Naval Research Laboratory
Washington, DC 20375-5320

NRL/MR/6400--01-8553

CHEMEQ2: A Solver for the Stiff Ordinary Differential Equations of Chemical Kinetics

David R. Mott
Elaine S. Oran

Laboratory for Computational Physics and Fluid Dynamics

July 27, 2001

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 27, 2001	

4. TITLE AND SUBTITLE

CHEMEQ2: A Solver for the Stiff Ordinary Differential Equations of Chemical Kinetics

5. FUNDING NUMBERS

6. AUTHOR(S)

David R. Mott and Elaine S. Oran

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/6400--01-8553

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

DOD Washington Headquarters Services
1155 Defense Pentagon, Room 3B269
Washington, DC 20301-1155

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This report describes and documents the subroutine CHEMEQ2, used to integrate stiff ordinary differential equations arising from reaction kinetics. This is a second generation improvement of CHEMEQ using a new quasi-steady-state predictor-corrector method that is A-stable for linear equations and second-order accurate. A single integration method can now be used for all species, regardless of the timescales of the individual equations. Start-up costs and memory requirements are low, so CHEMEQ2 is ideal for multi-dimensional reacting-flow simulations which require the solution of a process-split, initial-value problem in every computational cell for every global timestep. The algorithm, its implementation, the FORTRAN code, the internal variables and the argument lists are presented, along with several test problem results.

14. SUBJECT TERMS

Reacting flow
Stiff chemistry
ODE Integrator
Process-split methods
Quasi-steady state

15. NUMBER OF PAGES

67

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

Contents

CHEMEQ2: A Solver for the Stiff Ordinary Differential Equations of Chemical Kinetics

1 Introduction

This report documents CHEMEQ2, the latest version of the FORTRAN package CHEMEQ [1,2]. The CHEMEQ2 routines integrate sets of coupled, nonlinear ordinary differential equations (ODEs) of the form

$$\frac{dy_i}{dt} = g_i = q_i - p_i y_i, \qquad 1 \leq i \leq n, \tag{1}$$

where y_i is the density of the i^{th} species and g_i is its rate of change. Our primary application of Eq.(1) is to sets of coupled, nonlinear ODEs that represent chemical reaction sets. In this case, the dependent variables $\{y_i\}$ are concentrations or densities of reacting chemical species. Sometimes this equation is supplemented by another equation for the change in temperature or energy release that results from the species' interactions. The source term g_i, which is a function of the concentrations and the thermodynamic state, may be written as the difference of the production rate q_i and the loss rate $p_i y_i$. The timescales $\tau_i \equiv 1/p_i$ for the various species differ by many orders of magnitude and there may be strong coupling between species (i.e., the Jacobian matrix $\partial g_i/\partial y_j$ has significant off-diagonal elements). Under these circumstances, the set of equations represented by Eq. (1) is considered *stiff* and does not lend itself readily to numerical solution by classical methods such as the low-order Euler methods or higher-order Adams-Moulton methods [3–5]. Such a system then requires special techniques to obtain an accurate solution efficiently.

The coupled reaction set represented by Eq. (1) is often a part of a larger model that solves these equations coupled to the partial differential equations describing fluid dynamics. In such cases, chemical reactions are only one of several processes that might, for example, include advection, diffusion, or radiation transport. The numerical methods commonly used to solve such chemically reacting flows use *process splitting* (or *operator splitting*) [5]. The basic idea in operator splitting is to calculate the effects of individual physical processes separately for a chosen global timestep Δt_g, and then combine the results in some way. Each process in turn can change different system variables during Δt_g. Then, when it is time to integrate the ODEs representing the chemical changes during Δt_g, the integrator is presented with a new initial value

Manuscript approved May 9, 2001.

problem in each computational cell. The integrator must therefore solve

$$\frac{dy_i}{dt} = g_i, \quad y_i(t^0) = y_i^0 \qquad 1 \leq i \leq n, \tag{2}$$

to $t = t^0 + \Delta t_g$. The ODE integration may subdivide Δt_g into smaller steps, Δt, to obtain an accurate, stable solution. Here, the timestep Δt is called the *chemical timestep* because it is the timestep that the ODE integrator uses to advance the chemical reactions. The size of Δt generally varies during the course of the calculation.

Given that fluid dynamic calculations are seldom accurate to better than a few percent, any requirement of the chemical integrator to calculate the species concentrations more accurately than a few tenths of a percent is usually excessive. Therefore, the chemical integrator may be relatively low-order. Also, since the integrator must solve multiple initial value problems "from scratch" at every global timestep, it is necessary to use a single-point method, requiring information only from the current time level to calculate the concentrations at Δt. This is in contrast to multi-point methods that must store concentration or source-term values from several successive timesteps in order to advance the solution. Within the calculation for a given Δt_g, multi-point methods have a start-up penalty until a sufficient number of steps have been taken to build the history required for the calculation, and they often require interpolation procedures if Δt changes during the integration. The chemistry integration from the previous Δt_g does not provide the history needed to restart the integration because the fluid dynamics calculation changes the state. The values at the end of the previous chemistry integration are therefore not the values at the start of the next chemistry integration. By comparison, a single-point method has minimal start-up penalty at the beginning of an integration step and there is never a fluid dynamic inconsistency.

CHEMEQ [1,2] is a second-order single-step ODE integrator that has been used successfully as a part of a number of different types of reacting-flow codes. These have included applications to combustion [6–10] and solar physics [11–13]. CHEMEQ is a *hybrid method*, which means it chooses between a stiff method and a non-stiff method for integrating each ODE within the system depending upon the timescale of that equation. CHEMEQ has been shown to outperform standard stiff ODE solvers by a factor of 50–100 in speed in validation studies on chemical integrations alone (i.e., not coupled to fluid dynamics) when only moderate

accuracy was required [1]. More recently, an integrator based heavily on CHEMEQ outperformed a first-order quasi-steady-state method and the implicit preconditioning method CHEMSODE [30] on a photochemical smog problem [29]. Despite its strengths, CHEMEQ exhibits instability under some situations and is limited in the accuracy it can achieve [19].

This report describes a quasi-steady-state method which we call α-QSS, and its implementation in the subroutine CHEMEQ2. The α-QSS method is A-stable for linear problems and second-order accurate. It is more stable, more accurate, and costs less than CHEMEQ, and it successfully integrates some systems for which CHEMEQ fails [19]. CHEMEQ2, has been used successfully in hydrogen-air flame studies in microgravity [16], on pulse-detonation engine studies [17, 18], on thermonuclear mechanisms used in supernova simulations, and on the test cases used to validate CHEMEQ [19]. In addition to describing the new algorithm, we present error and linear stability analyses. We also describe how to use the subroutine CHEMEQ2 as a stand-alone integrator and in conjunction with a reacting-flow code. Variables used in CHEMEQ2 are listed and documented in Appendix A, and results obtained using CHEMEQ2 are compared to those obtained using CHEMEQ on two test problems. Finally, code listings for CHEMEQ2 and its supporting subroutines are also included.

2 Introduction to QSS Methods

Consider a simplified form of Eq. (1), in which the subscript i is dropped for convenience, $t^0 = 0$, and $y(t^0) = y^0$,

$$\frac{dy}{dt} = q - p\,y \qquad y(0) = y^0. \tag{3}$$

If p and q are constant, then Eq. (3) has an exact solution given by

$$y(t) = y^0 e^{-p\,t} + \frac{q}{p}\left(1 - e^{-p\,t}\right). \tag{4}$$

Quasi-steady-state (QSS) methods are based on the solution given in Eq. (4) [21–24]. If q and p are slowly varying, evaluating Eq. (4) at $t = \Delta t$ using $q(t^0)$ and $p(t^0)$ provides a good approximation for $y(\Delta t)$. This approach gives a first-order method which is the simplest QSS algorithm. More sophisticated QSS algorithms incorporate the time dependence of p and q and may place Eq. (4) into an alternate algebraic form. The

common thread between QSS methods is their basis on Eq. (4), which requires the methods to return the exact solution if q and p are constant. There are many QSS methods documented in the literature, and the α-QSS method is compared to several of them in Section 3.2.

Note that if $p\Delta t \rightarrow \infty$ (i.e., the ODE timescale is very small compared to the numerical timestep), the exponential terms in Eq. (4) may be neglected completely, leading to the *steady-state approximation* [31]

$$y \approx \frac{q}{p}. \tag{5}$$

Since q and p are functions of t, a steady-state approximation for species i does not imply that the value of y_i remains constant. Equation (5) assumes that the adjustment toward a "local" equilibrium for species i based on the current values of q_i and p_i is instantaneous. In contrast, quasi-steady-state methods retain information about the transition to equilibrium and are therefore more accurate for larger timescales. Note that some authors call Eq. (5) a quasi-steady-state approximation [32], emphasizing the continued evolution of y_i as q_i and p_i change. In this paper, the label "quasi-steady state" is reserved for methods that reproduce the solution in Eq. (4) for constant q and p regardless of the timescale.

QSS methods are often compared with standard stiff solvers such as LSODE [25,26], which is a variable-order method based on Gear's backward differentiation formulae (BDF) [27]. However, such comparisons have been largely limited to the integration of a single problem from one set of initial conditions, not reacting-flow simulations in which start-up overhead and storage requirements play key roles in the overall efficiency of the integrator. Verwer and Simpson describe one such test from atmospheric chemistry, in which a simple two-step BDF method outperforms a first-order implicit QSS method and a two-stage explicit QSS method. The test involved the calculation of emissions and was not coupled to fluid dynamics [22]. Jay et al. introduce two QSS methods and examine their performance on a set of atmospheric tests involving 32 species [21]. These two QSS methods outperformed both a standard, first-order QSS method and CHEMEQ, but the methods were slower than multi-point BDF methods. Variable-order, multi-point BDF methods often outperform QSS methods when the chemistry integration stands alone. However, the demands of a reacting-flow application are very different than those of a stand-alone integration, and the conclusions of these studies cannot be applied to reacting-flow problems. The α-QSS algorithm was developed specifically to meet the

demands of a process-split, reacting-flow simulation.

3 The α-QSS Algorithm

3.1 Algorithm Development

Given the demands of a reacting-flow application, we chose a predictor-corrector implementation for the integrator. Evaluating Eq. (4) at Δt using initial values serves as the predictor step, and a correction based on the initial and the predicted values then follows. The corrector step can be repeated using the previous corrector result as the new predicted value. Predictor-corrector methods of this type are single-point methods because information from only a single time level is needed to initiate calculation of the solution at the next time level.

First, a convenient algebraic form for Eq. (4) was chosen. Equation (4) can be evaluated at $t = \Delta t$, yielding

$$y(\Delta t) = y^0 + \frac{\Delta t(q - p\,y^0)}{1 + \alpha p \Delta t}, \tag{6}$$

for α defined by

$$\alpha(p\,\Delta t) \equiv \frac{1 - (1 - e^{-p\,\Delta t})/(p\,\Delta t)}{1 - e^{-p\,\Delta t}}. \tag{7}$$

The parameter α is a function of $p\,\Delta t$, as shown in Fig. 1. Note that $\alpha \to 0$ as $p\,\Delta t \to -\infty$, $\alpha \to 1$ as $p\,\Delta t \to \infty$, and $\alpha = 1/2$ for $p\,\Delta t = 0$. The meanings of these limits are clarified by recalling that $p\,\Delta t = \Delta t/\tau$. The $\alpha \to 1$ limit corresponds to an infinitely fast ODE relative to Δt, and $\alpha = 1/2$ corresponds to an infinitely slow ODE. Equation (6) is exact for any value of p (provided q and p are constant). However, we split g such that $p\,y$ is a non-negative loss rate, so only values of $p\,\Delta t \geq 0$ need be considered. Approximations used to calculate $\alpha(p\Delta t)$ without the costly exponential evaluation are described in Section 4.1.

A predictor-corrector method based on the solution in Eq. (6) takes the form

$$y^p = y^0 + \frac{\Delta t\,(q^0 - p^0 y^0)}{1 + \alpha^0\,\Delta t\,p^0} \qquad \text{Predictor,} \tag{8}$$

$$y^c = y^0 + \frac{\Delta t\,(q^* - p^* y^*)}{1 + \alpha^*\,\Delta t\,p^*} \qquad \text{Corrector.} \tag{9}$$

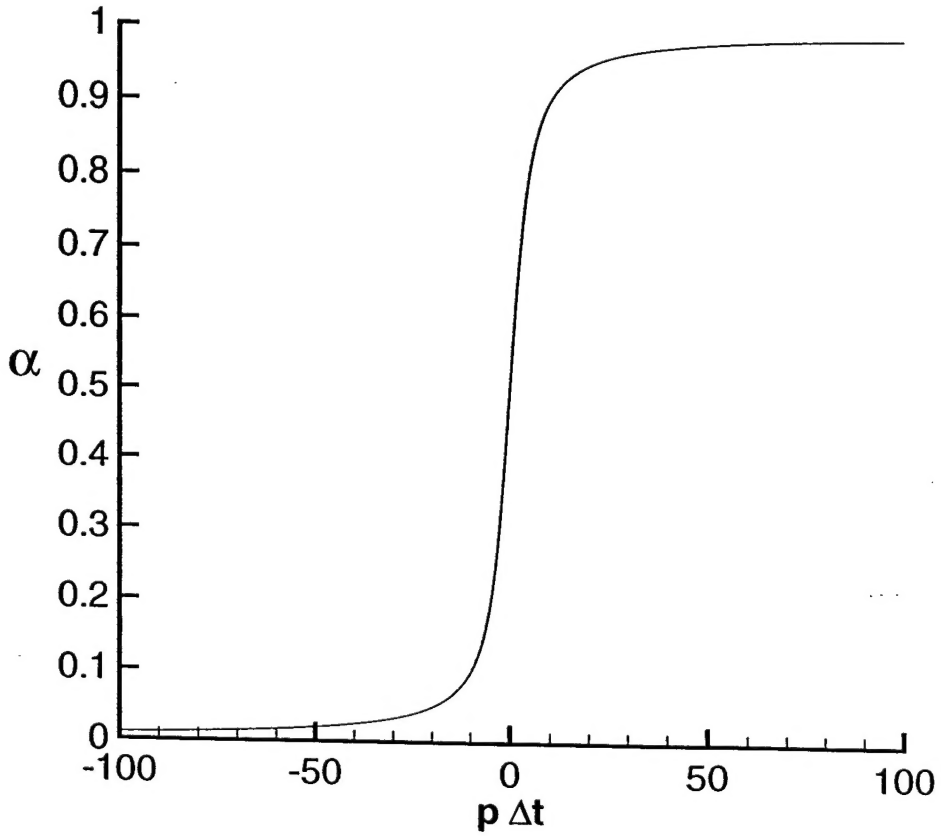

Figure 1: The parameter α as a function of $p\Delta t$.

Superscript 0 indicates initial values, and superscripts p and c indicate predicted and corrected values, respectively. The predictor uses the initial values of q, p, and y, but the "starred" variables (q^*, p^*, y^*, and α^*) can be based on both the initial values and the predicted values.

If we assume linear profiles in time for q and p between the initial and predicted values, we can find an exact series solution for Eq. (1). (This solution is illustrated in conjunction with the error analysis in Section 3.3.) Unfortunately, the series solution does not readily provide an efficient integration technique, nor does it indicate appropriate averages for the starred variables in the corrector. However, solutions do exist under slightly simpler conditions that can be reproduced with appropriate choices of the starred variables.

For instance, if p is constant and q is linear in time, the exact solution to Eq. (1) can be written as

$$y(\Delta t) = y^0 + \frac{\Delta t\left(\widetilde{q} - py^0\right)}{1 + \alpha\,\Delta t\,p}, \tag{10}$$

for $\alpha = \alpha(p\Delta t)$ from Eq. (7) and

$$\widetilde{q} = \alpha q(\Delta t) + (1-\alpha)q^0. \tag{11}$$

Alternatively, if $q = 0$ and p is linear in time, the exact solution of Eq. (1) is

$$y(\Delta t) = y^0 + \frac{\Delta t\left(-\overline{p}y^0\right)}{1 + \overline{\alpha}\,\Delta t\,\overline{p}}, \tag{12}$$

in which

$$\overline{p} = \frac{1}{2}\left(p(\Delta t) + p^0\right), \tag{13}$$

and $\overline{\alpha} = \alpha(\overline{p}\Delta t)$ from Eq. (7).

These results suggest a corrector of the form

$$y^c = y^0 + \frac{\Delta t\left(\widetilde{q} - \overline{p}y^0\right)}{1 + \overline{\alpha}\,\Delta t\,\overline{p}}. \tag{14}$$

To calculate $\widetilde{q}$ and $\overline{p}$ from Eqs. (11) and (13), we replace $q(\Delta t)$ and $p(\Delta t)$ with the predicted values q^p and p^p. When q and p are known functions of t, the predicted values are replaced with the exact values at Δt. Using $\{\widetilde{q}, \overline{p}, y^0, \overline{\alpha}\}$ for $\{q^*, p^*, y^*, \alpha^*\}$ in Eq. (9) gives a method which is A-stable for linear problems and second-order accurate, as is shown in Sections 3.3 and 3.4. We refer to the new method as α-QSS, which

emphasizes the dual role that α plays in returning the exact solution for constant q and p and in providing a weighted average of q when q is not constant.

3.2 Comparison to Previous Methods

In addition to the algebraic form chosen for Eqs. (8) and (14), α-QSS differs from previous QSS methods in its choice of averaging and its implementation as a predictor-corrector method. Previous methods that calculate average values for p and q use the same averaging method for both terms. For example, the two-stage explicit method introduced by Verwer and Van Loon [23] and tested by Verwer and Simpson [22] uses a simple algebraic average for both q and p calculated from initial and predicted values. CREK1D [24] uses an implicit exponential Euler formulation in which $\alpha(p\Delta t)$ gives a weighted average of the *composite* source terms:

$$y(\Delta t) = y^0 + \Delta t \left(\alpha g(\Delta t) + (1-\alpha) g^0\right). \tag{15}$$

In contrast, the α-QSS algorithm uses a simple algebraic average for p and an α-weighted average for q in order to match the exact solutions described in Eqs. (10) and (12).

Other QSS methods combine the results of first-order calculations in a way that improves accuracy. Jay et. al. [21] describes two such methods. Their "extrapolated QSS" method finds the solution at $t^0 + \Delta t$, first with a single step and then with two steps of $\Delta t/2$ each. A simple extrapolation then estimates the solution that would result if an infinitely small timestep were used. Their second method, "symmetric QSS," is a two-step method requiring three evaluations of the source terms. Each of these steps acts as if q and p were constant, and the values for q and p are taken at the same time level based on the previous calculation. No averaging of q or of p occurs between time levels in these methods.

The algebraic form of Eqs. (8) and (14), which was introduced in Eq. (6), is based on the asymptotic update employed by CHEMEQ when the timescale for an equation is smaller than some user-specified value [1,2]. However, CHEMEQ effectively replaces $\alpha(p\Delta t)$ with the constant $1/2$, which is equivalent to choosing the Padé approximation

$$\exp(x) \approx \frac{2+x}{2-x} \tag{16}$$

in either the definition of α or in Eq. (4). When the timescale for an equation is larger than some user-specified value, that equation is integrated using the modified Euler method. The hybrid method studied by Lorenzini and Passoni [29] uses CHEMEQ's update equations but different criteria for determining the timestep and for choosing between the asymptotic update and the modified Euler update. CHEMEQ's asymptotic update also uses different averages in the corrector for p and q than those used in Eq. (14). These differences lead to instability in CHEMEQ that is discussed more thoroughly in Section 6.1. The averages chosen by α-QSS eliminate this instability, and α-QSS automatically approaches the modified Euler method as $p\Delta t \rightarrow 0$.

3.3 Error Analysis

The method has a third-order error term for a single step, which makes it second-order over the course of an integration. This can be shown by examining the exact series solution of Eq. (1). Writing the series for $y(t)$ about $y(t^0 = 0) = y_0$,

$$y(t) = y_0 + y_1 t + y_2 t^2 + \ldots = \sum_{j=0}^{\infty} y_j t^j, \tag{17}$$

the derivative is given by

$$\frac{dy}{dt} = y_1 + 2 y_2 t + +3 y_3 t^2 + \ldots = \sum_{j=1}^{\infty} j y_j t^{j-1}. \tag{18}$$

This development deals with a single species, y, so y_j is the coefficient of the t^j term in the expansion in Eq. (17) and not the concentration of the j^{th} species in a multi-species system. Similarly, series expansions for $q(t)$ and $p(t)$ are given by

$$q(t) = \sum_{j=0}^{\infty} q_j t^j, \tag{19}$$

$$p(t) = \sum_{j=0}^{\infty} p_j t^j. \tag{20}$$

Substitution into Eq. (1) using Eqs. (17)-(20) gives

$$y_1 = q_0 - p_0 y_0, \tag{21}$$

$$y_2 = \frac{1}{2}\left(q_1 - (p_1 y_0 + p_0 y_1)\right), \tag{22}$$

$$y_3 = \frac{1}{3}\left(q_2 - (p_2 y_0 + p_1 y_1 + p_0 y_2)\right), \tag{23}$$

and leads to the general expression

$$y_j = \frac{1}{j}\left(q_{j-1} - \sum_{k=0}^{j-1} p_{j-1-k} y_k\right) \tag{24}$$

for $j > 0$. (The exact solution for q and p both linear in time that was mentioned in Section 3.1 is obtained by assuming $p_j = q_j = 0$ for $j \geq 2$.)

In general, q and p are given as functions of y, not as functions of t. Therefore, the coefficients in Eqs. (19) and (20) are not known, and Eq. (3) is a nonlinear differential equation. We will first perform an error analysis for the linear version of Eq. (3), in which q and p are known functions of t, and then extend this analysis for the nonlinear case. For the linear case, the predicted values are simply $q^p = q(\Delta t)$ and $p^p = p(\Delta t)$. Subtracting the series expansion for Eq. (14) from the exact solution evaluated at $t = \Delta t$ yields

$$y(\Delta t) - y^c = \frac{\Delta t^3}{6}\left(-\frac{1}{2}p_1 q_0 - q_2 + p_2 y_0\right) + O(\Delta t^4) \qquad \text{[linear case]} \tag{25}$$

The leading error term is $O(\Delta t^3)$ per timestep. Since the number of timesteps required to reach a given time is proportional to $1/\Delta t$, the error for the method is second-order when these errors all reinforce [3]. Note that this error term disappears for constant p and linear q (which gives $p_1 = q_2 = p_2 = 0$) and linear p with $q = 0$ (which gives $q_0 = q_2 = p_2 = 0$). These two cases were used to choose how to calculate the starred variables in Eq. (9), and the method is exact for either case.

The method is second-order for nonlinear problems as well. To illustrate this, first note that the leading error term for the predicted values y^p is second-order:

$$y(\Delta t) - y^p = \frac{\Delta t^2}{2}(q_1 - p_1 y_0) + O(\Delta t^3). \tag{26}$$

Since q and p are polynomials in the species concentrations for the nonlinear systems representing reaction kinetics, the leading error terms for the predicted values q^p and p^p are also second-order. This error can be represented as

$$q(\Delta t) - q^p = \epsilon_q \Delta t^2 + O(\Delta t^3), \tag{27}$$

$$p(\Delta t) - p^p = \epsilon_p \Delta t^2 + O(\Delta t^3), \tag{28}$$

for some unknown coefficients ϵ_q and ϵ_p. Using these predicted values in Eq. (14) gives an error term of the form

$$y(\Delta t) - y^c = \Delta t^3 \left(-\frac{1}{12} p_1 q_0 - \frac{1}{6} q_2 + \frac{1}{6} p_2 y_0 + \frac{1}{2} \epsilon_q - \frac{1}{2} \epsilon_p y_0 \right) + O(\Delta t^4) \qquad \text{[nonlinear case]}. \tag{29}$$

As with the linear problem, the leading-order error term for the nonlinear problem is $O(\Delta t^3)$ per timestep, so the method is still at least second-order over the course of an integration.

3.4 Linear Stability Analysis

For the single linear equation

$$\frac{dy}{dt} = \lambda y, \tag{30}$$

the coefficient λ can be a function of t but not a function of y. Using the average value $\overline{\lambda}$ given by

$$\overline{\lambda} = \frac{1}{2} \left(\lambda(t = 0) + \lambda(t = \Delta t) \right), \tag{31}$$

α-QSS has amplification factor G given by

$$G = 1 + \frac{\overline{\lambda} \Delta t}{1 - \overline{\alpha}\, \overline{\lambda} \Delta t}. \tag{32}$$

The signs in Eq. (32) reflect the fact that $\lambda = -p$, and note that $\overline{\alpha} = \alpha(-\overline{\lambda} \Delta t)$. Using Eq. (7), the expression for G simplifies to

$$G = \exp(\overline{\lambda}\, \Delta t). \tag{33}$$

For $\overline{\lambda} = a + b\sqrt{-1}$ with a, b both real, the magnitude of G is simply

$$\|G\| = \exp\left(a\, \Delta t\right). \tag{34}$$

Since $\|G\| \leq 1$ for $a \leq 0$ for any value of b, the method is A-stable. This does not prove that α-QSS is A-stable when applied to nonlinear systems of ODEs for which $\{p_i\}$ and $\{q_i\}$ depend on $\{y_i\}$. However, in testing to date, the accuracy-based timestep criterion used originally in CHEMEQ has worked well for α-QSS. To ensure that stability is maintained when the corrector is iterated, a new stability check was introduced [20] for Δt. These accuracy and stability criteria are discussed in Section 4.2.

4 CHEMEQ2 Implementation

4.1 Update Equations

CHEMEQ2 uses the α-QSS update on all equations in the system regardless of the timescale of the ODE. This makes the timestep less prone to oscillations than for hybrid methods (like CHEMEQ) that switch between update methods. In addition, iterations may be done on the corrector that improve the accuracy of the result.

Again using a superscript 0 to indicate values at the begining of the chemical timestep and a subscript i to specify species i, the α-QSS update is given by

$$y_i^p = y_i^0 + \frac{\Delta t\,(q_i^0 - p_i^0\, y_i^0}{1 + \alpha_i^0 \Delta t\, p_i^0} \qquad \text{Predictor,} \tag{35}$$

$$y_i^c = y_i^0 + \frac{\Delta t\,(\widetilde{q_i} - \overline{p_i}\, y_i^0)}{1 + \overline{\alpha_i} \Delta t \overline{p_i}} \qquad \text{Corrector.} \tag{36}$$

The predictor uses all initial values, and $\alpha_i^0 = \alpha(p_i^0 \Delta t)$. After calculating the predicted concentrations $\{y_i^p\}$ for all of the species in the system, next obtain $\{q_i^p\}$ and $\{p_i^p\}$ from $\{y_i^p\}$. Then calculate

$$\overline{p_i} = \frac{1}{2}\left(p_i^0 + p_i^p\right), \tag{37}$$

evaluate $\overline{\alpha_i} = \alpha(\overline{p_i}\Delta t)$, and finally

$$\widetilde{q_i} = \overline{\alpha_i} q_i^p + (1 - \overline{\alpha_i}) q_i^0. \tag{38}$$

These averages are then used to calculate y_i^c, and to iterate on the corrector, use the value for y_i^c from one step as y_i^p for the next step.

Having an accurate approximation for $\alpha(p\Delta t)$ that does not require an evaluation of the exponential function makes the method given by Eqs. (35) and (36) more attractive. Recall that p is strictly non-negative based on the way the chemical source term is split, so this approximation need only hold for positive values of $p\,\Delta t$. Equation (7) indicates that as $p\Delta t \to \infty$, a reasonable approximation for $\alpha(p\,\Delta t)$ is

$$\alpha(p\,\Delta t) \approx 1 - \frac{1}{p\,\Delta t}. \tag{39}$$

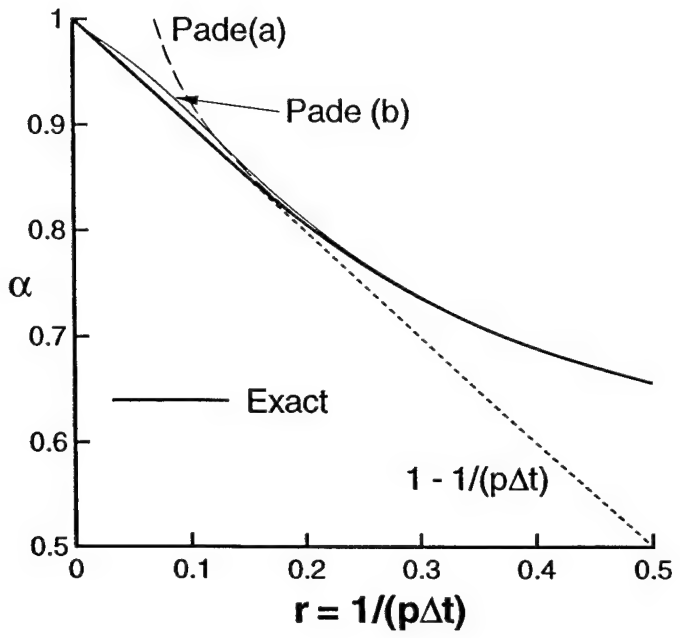

Figure 2: Approximations for α as a function of $r = 1/(p\Delta t)$.

Using this approximation for α eliminates the need to find an accurate approximation for e^{-x} as $x \to \infty$, as would be required if the solver were based on Eq. (4) rather than Eq. (6). Using the Padé approximation [4]

$$e^x \approx \frac{1680 + 840x + 180x^2 + 20x^3 + x^4}{1680 - 840x + 180x^2 - 20x^3 + x^4} \tag{40}$$

in the definition of $\alpha(p\,\Delta t)$ gives

$$\alpha(\Delta t/\tau) \approx \frac{840r^3 + 140r^2 + 20r + 1}{1680r^3 + 40r} \tag{41}$$

for $r = 1/(p\,\Delta t)$. These two approximations are shown with the exact curve for α in Fig. 2. The approximation given by Eq. (41) is labeled Padé (a). Note that unlike Fig. 1, the x-coordinate in Fig. 2 is $\tau/\Delta t$. The linear approximation in Eq. (39) is closer to the exact value of α than the approximation in Eq. (41) for $r \leq 0.16762$; for $r > 0.16762$, Eq. (41) is more accurate. Therefore, the better approximation can be chosen based upon the value of r. This combined approximation differs from the exact value of α by at most 0.3%. This error occurs in a narrow region around the transition from the linear to the rational approximation. As $r \to \infty$, Eq. (41) is actually better behaved numerically than Eq. (7), which requires the exponential function evaluation.

A second approximation for α that can be used alone (i.e., does not require the linear approximation, Eq. (39), to take over as $r \to 0$) results from the Padé approximation

$$e^x \approx \frac{360 + 120x + 12x^2}{360 - 240x + 72x^2 - 12x^3 + x^4}. \tag{42}$$

This approximation gives

$$\alpha(\Delta t/\tau) \approx \frac{180r^3 + 60r^2 + 11r + 1}{360r^3 + 60r^2 + 12r + 1}. \tag{43}$$

This second approximation for α is designated Padé (b) and is also shown in Fig. 2. Since this approximation recovers $\alpha = 1$ for $\tau/\Delta t = 0$, Eq. (43) can be used alone with only a slight accuracy penalty compared to the combined approximation of Eqs. (39) and (41). In testing to date, this accuracy penalty in α has not caused accuracy problems in the species solutions, and using this single equation eliminates the logic check required to determine which of Eqs. (39) or (41) to use for the combined method. The success of CHEMEQ, which effectively uses $\alpha = 1/2$, suggests that an even simpler approximation for $\alpha(p\Delta t)$ than those shown here may provide sufficient accuracy with lower computational cost.

4.2 Timestep Selection

Accuracy is controlled by choosing Δt and the number of corrector iterations, N_c. A single corrector calculation is performed if $N_c = 1$, and as Appendix B illustrates, increasing N_c improves accuracy. Timestep selection is identical to that used by CHEMEQ [2]. The initial predicted values and final corrected values are tested to see if they satisfy

$$\|y_i^c - y_i^p\| \leq \epsilon\, y_i^c \tag{44}$$

for some specified constant ϵ, typically $\sim 10^{-2}$. If Eq. (44) is not satisfied, the step is repeated with a smaller timestep. As Young notes [2], it is best to reduce the timestep sharply (a factor of 2 or 3) rather than slowly as less time is wasted finding a step size sufficiently small for convergence. However, if the convergence criterion is met easily during the iteration, it is best to only increase the step size by 5–10% each step [2].

The timestep modification is performed by modeling the difference between predictor and corrector as a

single second-order term. Choose a_2 such that

$$y_i^c - y_i^p = a_2(\Delta t)_{old}^2, \tag{45}$$

where $(\Delta t)_{old}$ is the timestep used to calculate y_i^p and y_i^c from the initial conditions. The user specifies a target value for the relative magnitude of this correction term, given by

$$\|a_2 \Delta t^2\|_{target} = \varepsilon\, y_i^c. \tag{46}$$

The values of ϵ in Eq. (44) and ε in Eq. (46) are related by

$$\epsilon = c\varepsilon \tag{47}$$

for some user-specified $c > 1$, so the error criterion in Eq. (44) is rarely violated for the chosen timestep. Defining the parameter σ as

$$\sigma \equiv \max_i \left(\frac{\|y_i^c - y_i^p\|}{\varepsilon\, y_i^c} \right), \tag{48}$$

the value of Δt that limits the largest relative concentration change to the target magnitude is

$$(\Delta t)_{target} = \frac{(\Delta t)_{old}}{\sqrt{\sigma}}. \tag{49}$$

Again, the difference $\|y_i^c - y_i^p\|$ is calculated using the initial prediction for y_i^p and the final corrected value of y_i^c.

In CHEMEQ2, the new timestep is calculated using

$$(\Delta t)_{new} = (\Delta t)_{old} \left(\frac{1}{\sqrt{\sigma}^*} + .005 \right). \tag{50}$$

In Eq. (50), $\sqrt{\sigma}^*$ is an estimate by three Newton iterations for $\sqrt{\sigma}$ using σ as the starting value. Equation (50) gives the desired asymmetrical property in that Δt decreases faster than Δt would increase for the inverse value of σ. In addition, Δt is modified very little when σ is near 1.

If Eq. (44) is satisfied, the new concentration values at $t = t^0 + (\Delta t)_{old}$ are set equal to the values of y_i^c and the integration continues with timestep $(\Delta t)_{new}$. A successful integration step requires only two derivative function evaluations when a single corrector step is used, and the timestep selection algorithm minimizes the number of steps that must be repeated.

When y_i is decaying toward zero, it is constrained by a minimum value. When this lower bound is reached, the species is no longer considered in the calculation of σ and therefore does not affect convergence. The lower bound is chosen by the user and must be selected to insure that it does not adversely affect the accuracy of the solution. If the minimum value is too high, a species value could freeze prematurely. This could corrupt the solution for other species whose values are sensitive to the frozen species. If the minimum is too low, it can affect efficiency if the decaying mode controls the timestep. Thus it is better to choose minimum values that are too small rather than too big. This may slightly reduce computational efficiency, but it will also reduce the possibility of calculating an incorrect solution.

As mentioned in Section 3.4, α-QSS is A-stable for linear problems, but this result holds no guarantees for the nonlinear systems of chemical kinetics. To ensure that the calculation remains stable, the integrator can monitor the convergence of the corrector iteration and adjust the timestep if necessary. Let $y_i^{c(l)}$ denote the corrected value of $y_i(\Delta t)$ after l iterations. The change from one iteration to the next,

$$\Delta y_i^{c(l)} = y_i^{c(l)} - y_i^{c(l-1)} \tag{51}$$

should decrease in size as l grows if the iteration is stable. Therefore, requiring that

$$\|\Delta y_i^{c(N_c)}\| < \|\Delta y_i^{c(N_c-1)}\|, \tag{52}$$

where N_c is the specified maximum number of iterations, ensures that the integration remains stable. A step using Δt that satisfies the accuracy constraint but fails to satisfy Eq. (52) for any i is repeated using a new timestep, $(\Delta t)_{new}$, given by

$$(\Delta t)_{new} = \Delta t \max_i \left(\frac{\|\Delta y_i^{c(N_c-1)}\|}{\|\Delta y_i^{c(N_c)}\| + 0.001} \right). \tag{53}$$

This instability is generally not seen in reacting-flow applications, so a more sophisticated criterion and timestep update have not been pursued. The constraint is available in CHEMEQ2, however, should it be needed.

At start-up, an initial timestep is chosen which approximates the timestep that will be determined by the predictor-corrector scheme. This initial trial timestep is determined such that none of the variables will change by more than a prescribed amount. If the formation rate q_i is much larger than the loss rate $p_i\, y_i$,

it is reasonable to assume that q_i and p_i will remain relatively constant for large changes in y_i. The initial change in y_i may be large enough to equilibrate the formation and loss rates for y_i. Thus the initial trial timestep is chosen as

$$\Delta t = \epsilon \min\left(\frac{y_i}{\dot{y}_i}, \text{ or (if } q_i \gg p_i y_i)\, 1/p_i\right), \tag{54}$$

where ϵ is a scale factor that need not be identical to the constant used in Eq. (44). Equation (54) is used only once per global timestep, as subsequent timesteps taken until the end of the global timestep are determined using Eq. (50).

The effect of the thermodynamic state on the the reaction rate constants has been ignored in the previous developments. The rate constants are often calculated once before the predictor step using the initial values and held constant during the corrector step. A new thermodynamic state is then found at the new time level and used for the following predictor and corrector. If the integration is particularly sensitive to the thermodynamic state, this state can be recalculated for the corrector using the predicted solution. If the system requires the integration of a thermodynamic variable (such as temperature) along with the species concentrations, then the source term for this extra variable is split just as with the concentrations. If there is no "loss" term for that variable that can be assumed proportional to the variable, then the entire source term is assigned to q, and the method reduces to the modified Euler method for that equation since $\alpha(0) = 1/2$.

5 How to Use CHEMEQ2

The following sections describe what a user must know about CHEMEQ2 in order to use the subroutine effectively. A description of the four entrance points for the subroutine and the argument variables used for each is included, as are diagrams indicating the calling sequence used in a stand-alone integration and in a reacting-flow application. Appendix A describes all internal variables, and Appendix B provides code listings.

CHEMEQ2 is designed as a replacement for CHEMEQ, so most of the original CHEMEQ code is retained in CHEMEQ2. The overall logical structure from CHEMEQ is retained, with the hybrid method replaced with α-QSS. Minor changes in input/output have also been implemented, and these are discussed in the

following section.

5.1 Entry Points

1. `CHEMEQ2(dtg,gsub,n,y)` is the main entrance point and is used to advance the chemical variables in time.

   - `dtg`: time for which the integration is performed; Δt_g of Section 1.
   - `gsub`: name of the derivative function evaluator subroutine that provides the source term as q and py. The form of gsub and its arguments are given in Section 5.2.
   - `n`: then number of equations to be integrated
   - `y`: array which holds the initial values at the start of the integration and returns the final values at the end of the integration

2. `CHEMSP(epsmn, epsmx, dtmn, tnot, itermx, ns, ymn, prt)` provides a means to set the solution parameters used the next time CHEMEQ is called. If the passed variable has value ≤ 0, then the default value built into the subroutine is used. If the passed value is > 0, then the corresponding parameter is set to the passed value. For a typical calculation in which the same solution parameters are used throughout the domain, this routine may be called only once to initialize these parameters. If the simulation involves multiple regions that make different demands on the speed or accuracy of the integration, then this routine may be called so that appropriate parameters are used in each region. The parameters set by each variable and the default value for these parameters are listed below. The distinction is made between the arguments of CHEMSP and the internal variables.

   - `epsmn`: sets `epsmin`, the accuracy parameter ε in Eq. (46) for determining the timestep. Default value of `epsmin`: 10^{-2}.
   - `epsmx`: sets `epsmax`, the parameter c in Eq. (47) for specifying when a step must be repeated using a smaller timestep. Default value of `epsmax`: 10.
   - `dtmn`: sets `dtmin`, the minimum allowed timestep. Default value of `dtmin`: 10^{-15}.

- `tnot`: sets `tstart`, the value of the independent variable at the start of the global timestep. Default value of `tstart`: 0.
- `itermx`: sets `itermax`, the number of corrector iterations performed. Default value of `itermax`: 1 (a single corrector step).
- `ns`: integer that indicates the number of entries in `ymin` to initialize with `ymn`
- `ymn`: sets `ymin`, the array which holds the minimum allowed values of the dependent variables (for integration control). Default value of `ymin(i)`: 10^{-20} for all values of i.
- `prt`: if `prt` = 0, the list of parameters set by the current call to CHEMSP is printed.

3. `CHEMCT(tmk)` provides diagnostics by printing the number of derivative function calls and the number of times an integration step was redone due to a violation of the accuracy criterion or the stability criterion.
   - `tmk`: `REAL` number used to identify the call; the value of the independent variable is often used.
4. `CHEMPR(y,n)` is called for diagnostic purposes and prints partial lists of internal variables. The variable definitions are the same as those in the CHEMEQ2 call.

5.2 Supporting Subroutines

1. CHEMER is a diagnostic routine which warns the user that the minimum timestep threshhold has been violated and that the integration has been stopped. No arguments are required. The user may supply additional error-checking capabilities or diagnostic output.
2. `gsub(y, q, d, t)` is the derivative function evaluator. The actual name of this subroutine is passed to CHEMEQ2 as an argument. This is so the routine can be changed in different regions or regimes.
   - `y`: array holding the dependent variables
   - `q`: production terms; entry `q(i)`= q_i in Eq. (1).
   - `d`: loss terms; entry `d(i)`= $p_i y_i$ in Eq. (1).
   - `t`: current value of the independent variable

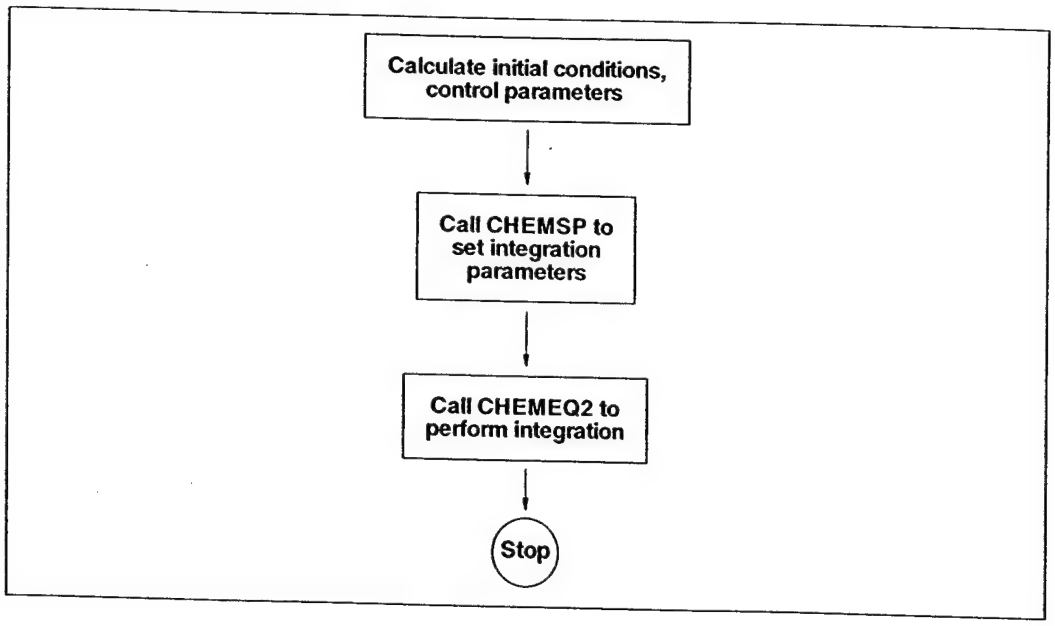

Figure 3: Basic calling sequence to obtain final values

5.3 Calling Sequence

If the user simply needs the species concentrations at t_{final} for a single problem, the basic calling sequence illustrated in Fig. 3 should be used. A single call to CHEMSP is required to initialize the integration parameters unless all default values are used, and $\mathtt{dtg} = t_{final}$. Using the version of CHEMEQ2 included in this report, no intermediate values are provided between $t = 0$ and t_{final}. The driver program included in Appendix C has this structure imbedded in a loop that scrolls through various solution parameters to give a set of final values. If intermediate values are desired, a write statement can be added within CHEMEQ2 in order to print the results of a successful step before the next step is taken. To obtain intermediate values without altering CHEMEQ2, the integration time t_{final} may be broken-up as illustrated in Fig. 4. After each Δt_g step, control returns to the driver program and values may be printed. The next step in the integration is taken, and the result of the previous step is used as the initial condition. The optional CHEMSP call is needed if the source terms depend on t (such as in atmospheric chemistry) or if the solution parameters are to be changed (i.e., perhaps the initial time interval is very sensitive and must be run more accurately, but later times allow this constraint to be loosened). Variables changed via CHEMSP and passed as arguments to CHEMEQ2 may be moved if necessary for efficiency. Please see the Section 5.4 for a discussion of some

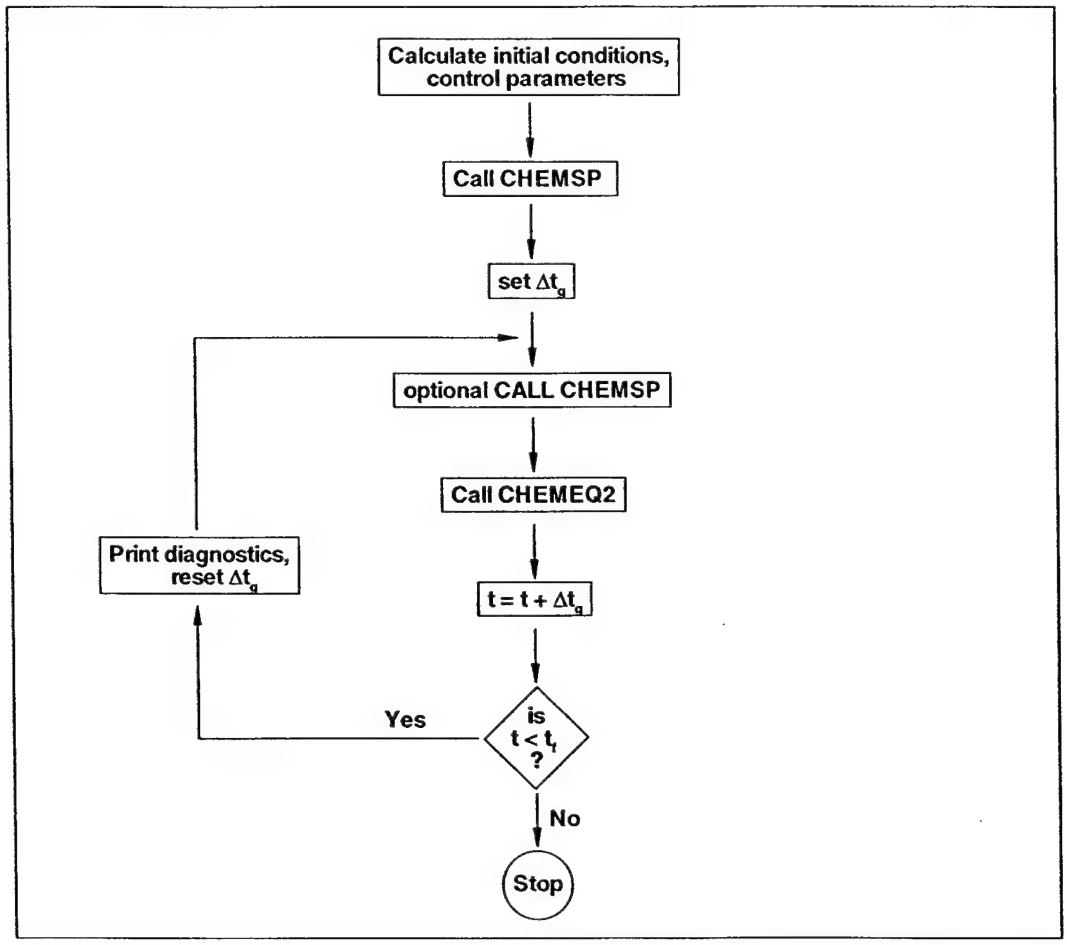

Figure 4: Calling sequence for a single-point integration that allows access to intermediate values.

practical aspects of using CHEMEQ2.

Figure 5 illustrates the use of CHEMEQ2 in a reacting flow code. The effects of the fluid dynamics are calculated by a separate algorithm, and then the conditions in each compuational cell are sent to CHEMEQ2 separately to calculate the effects of the chemistry. This is the simplest implementation for reacting flow, and as mentioned earlier massively parallel versions of CHEMEQ have been implemented [14, 15]. In general the optional calls to CHEMSP will not be needed since the solution parameters set by CHEMSP will be the same for all cells. If CHEMSP must be called repeatedly, the logical check that determines whether information about the call to CHEMSP is printed can be removed.

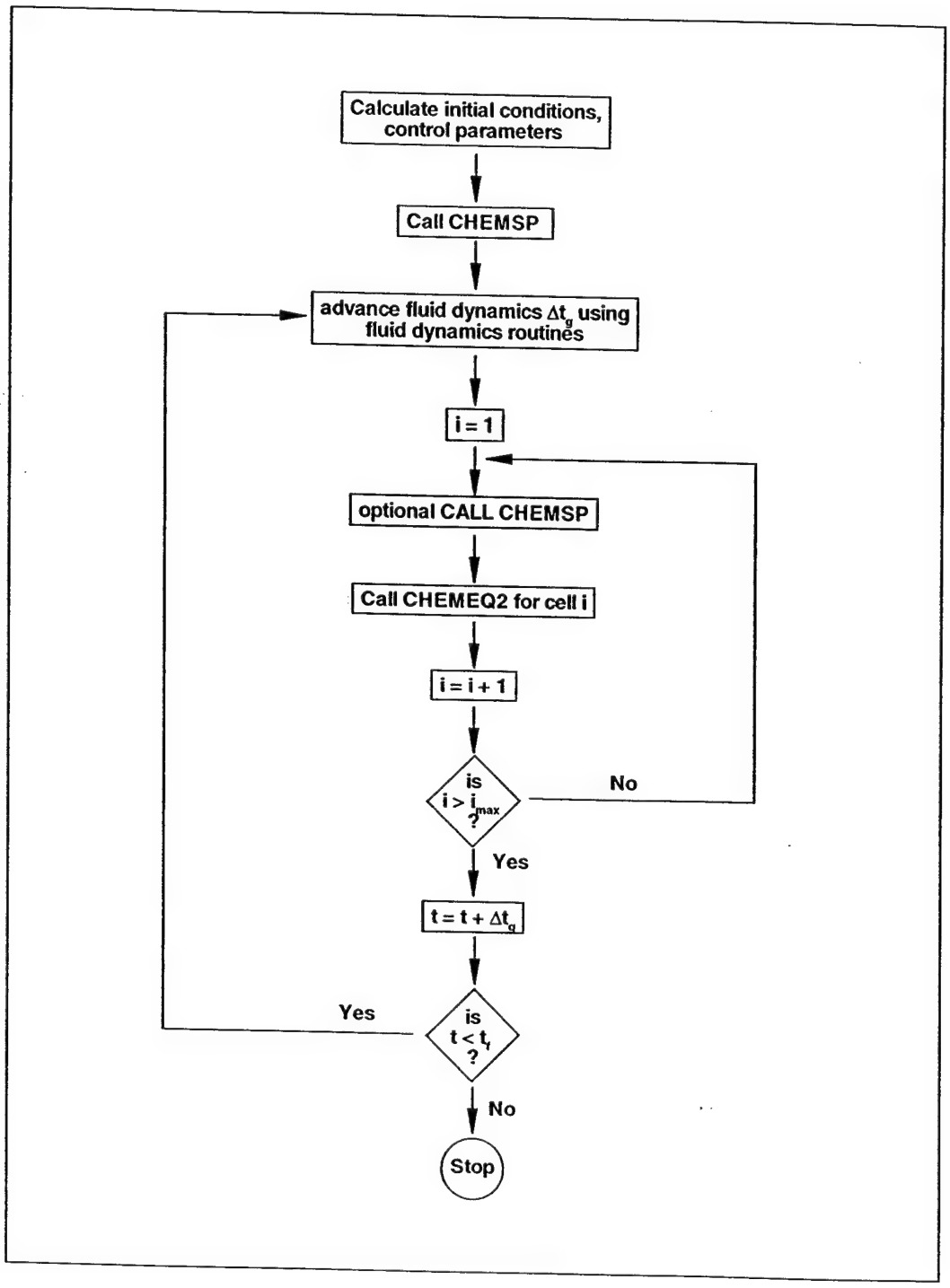

Figure 5: Calling sequence in a reacting-flow program. The parameter i_{max} gives the number of cells.

5.4 Practical Considerations

To help the beginning user of CHEMEQ2, some observations and suggestions are made for optimizing the use of the subroutine. First, the improvement produced by adding corrector iterations is problem specific. Some systems, such as cesium integration discussed in Section 6.1, converge in about three iterations. Other systems, such as the hydrogen-air mechanism discussed in Section 6.2, take much longer. Lowering ε may be more effective in improving accuracy than increasing N_c depending upon the problem. If a very accurate result is required and the added computational cost can be tolerated, increasing the number of corrector iterations dramatically is very effective [19].

As stated earlier, α-QSS is not guaranteed to be stable for nonlinear systems. If instability is seen, the user can use the convergence-based stability check on Δt discussed in Section 6.2 if $N_c \geq 3$. The lines which implement this check have a "D" in the first column. Many compilers allow the user to include these lines in the compilation by specifying a compiler option such as `-d_lines`. Without such an option, these lines are treated as comments and not compiled. Since most reacting-flow problems will not require the stability check, this implementation is most efficient. If the stability check is needed on a platform that does not support such a compiler option, then the lines must be manually included.

The two options for approximating α described in Section 4.1 are included in the code, but the one labeled Padé (b) is recommended. Although the combination of Padé (a) in Eq. (41) and the linear approximation is Eq. (39) are closer to the exact curve for α (see Fig. 2), the single equation given in Eq. (43) provides comparable accuracy in the species concentrations and eliminates an expensive logic check.

Finally, users may find the variable groupings in the argument lists of CHEMEQ2 and CHEMSP inconvenient. The user may wish to move arguments from one list to the other, concentrating the parameters that change regularly in the CHEMEQ2 argument list, and relegating to CHEMSP those parameters that need not be changed after an initialization. This could save the additional calls to CHEMSP to reset a single variable. For instance, if the source term calculation depends upon the value of the independent variable t, then this variable could be passed through the CHEMEQ2 argument list rather than set through a call to CHEMSP.

Table 1: Cesium Mechanism

Reaction		k_i
1)	$O_2^- + Cs^+ \xrightarrow{k_1} Cs + O_2$	5×10^{-8} cm^3/s
2)	$Cs^+ + e \xrightarrow{k_2} Cs$	1×10^{-12} cm^3/s
3)	$Cs \xrightarrow{k_3} Cs^+ + e$	3.24×10^{-3} s^{-1}
4)	$O_2^- \xrightarrow{k_4} O_2 + e$	4×10^{-1} s^{-1}
5)	$O_2 + Cs + M \xrightarrow{k_5} CsO_2 + M$	1×10^{-31} cm^6/s
6)	$O2 + e + O_2 \xrightarrow{k_6} O_2^- + O_2$	1.24×10^{-30} cm^6/s
7)	$O_2 + e + N_2 \xrightarrow{k_7} O_2^- + N_2$	1×10^{-31} cm^6/s

6 Numerical Results

Two examples are described here in detail. The first is a system of equations involving cesium and cesium ions that was originally suggested by D. Edelson of Bell Laboratories. This test was used to compare the original CHEMEQ subroutine to other stiff solvers, including those of Gear and Kregel, as shown in [2]. The second set of tests involves a hydrogen-oxygen combustion mechanism and focuses on the effect of corrector iteration on the timing and accuracy of α-QSS. Two reacting-flow applications are then discussed briefly in Section 6.3.

6.1 Cesium Tests

The cesium mechanism, shown in Table 1, involves seven species and seven one-way reactions. The rate constants k_i are fixed at the values shown. The inert collision partner, M in reaction 5, may be Cs, CsO_2, O_2, or N_2, so the concentration of M used to calculate the reaction rate is the sum of the concentrations of these four species. The initial conditions and the solution values at 1000 seconds used for the accuracy study are included in the Table 2 [2]. These solution values, which we call the "accepted values" in the following error analysis, are the common result of running LSODE and CHEMEQ2 at excessively high accuracies.

The species number densities, shown in Fig. 6, were generated using CHEMEQ2. The figure shows a fast initial transient, which is followed by a slow evolution toward equilibrium. A logarithmic scale in time is required to show this evolution. Equilibrium is not reached by 1000 s, so comparing the solution at this time to an accepted solution provides a suitable check of a kinetic integrator.

Table 2: Initial and $t = 1000$ sec species concentrations for the the cesium mechanism test problem.

Species	$y_i(0\,\mathrm{s})(\mathrm{cm}^{-3})$	$y_i(1000\,\mathrm{s})[\mathrm{cm}^{-3}]$
e	1×10^2	4.9657897283×10^4
O_2^-	5.2×10^2	2.5913949444×10^4
Cs^+	6.2×10^2	7.5571846728×10^4
Cs	1×10^{12}	1.5319405460×10^3
CsO_2	0	1.000×10^{12}
N_2	1.4×10^{15}	1.400×10^{15}
O_2	3.6×10^{14}	3.590×10^{14}

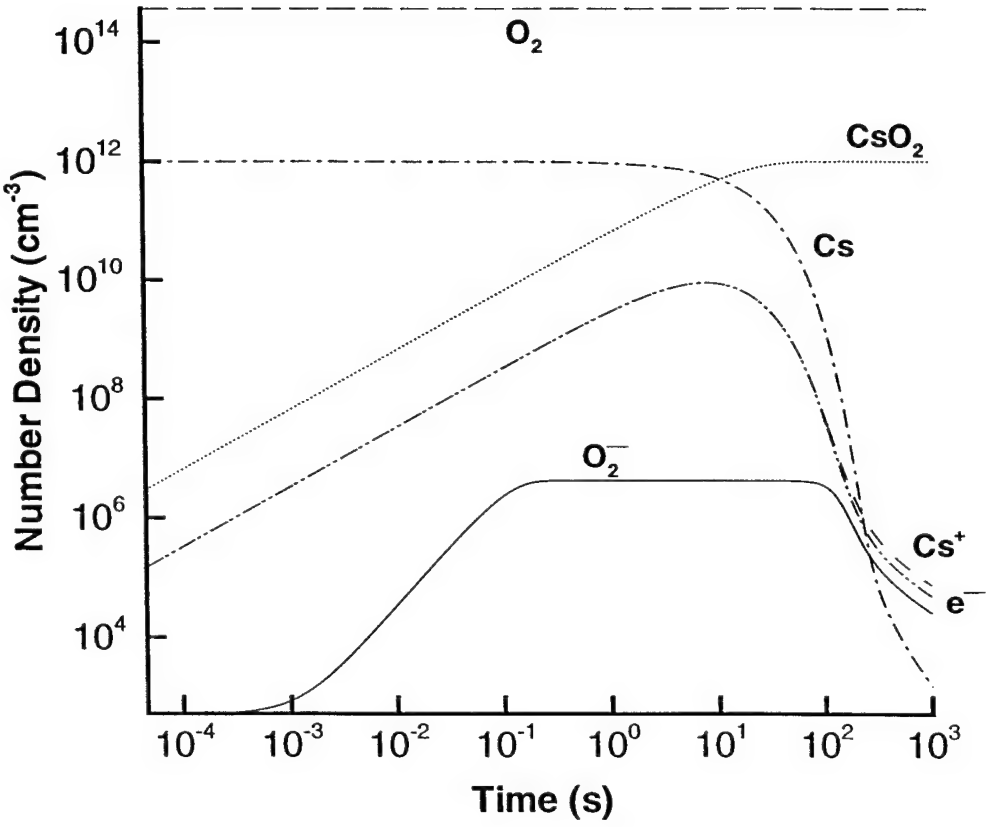

Figure 6: Species number densities as a function of time for the cesium mechanism test problem.

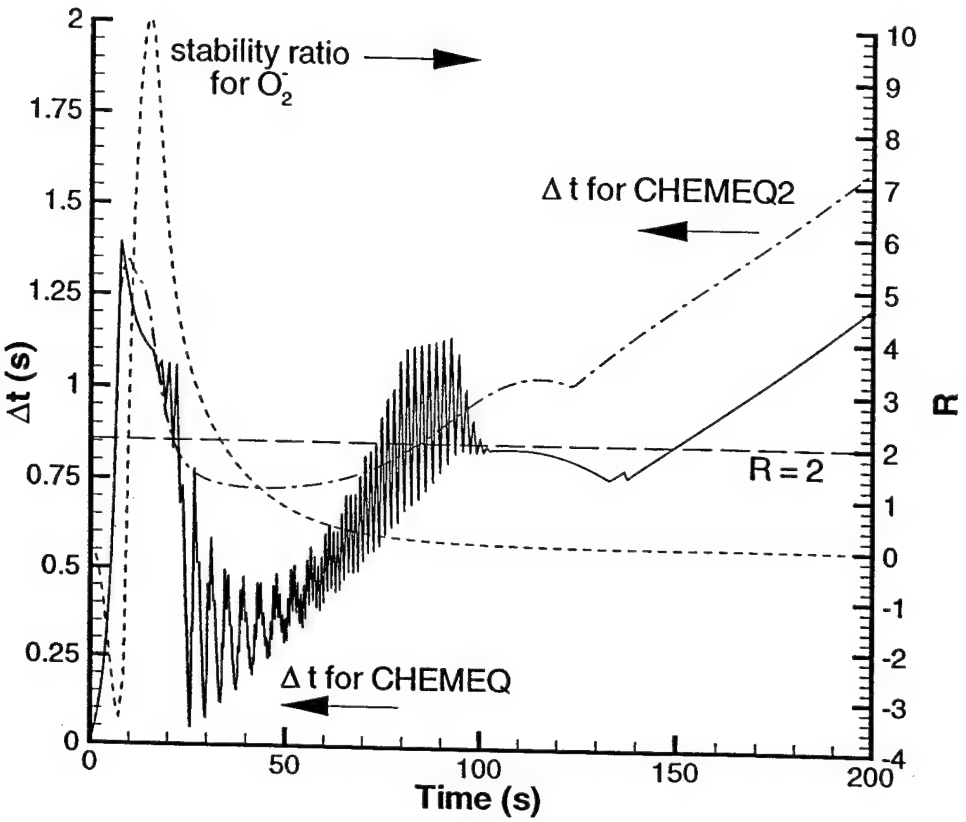

Figure 7: Timestep histories for the cesium integration using CHEMEQ and CHEMEQ2 for $\varepsilon = 0.01$, and the stability ratio, R, defined by Eq. (55), for O_2^-. R is calculated with Δt used by CHEMEQ2.

Timestep histories for CHEMEQ2 and for CHEMEQ are shown in Fig. 7. As mentioned earlier, the asymptotic update used by CHEMEQ is unstable under some circumstances [19]. The linear stability analysis of CHEMEQ led to a parameter R, defined by

$$R \equiv \frac{\Delta t}{\tau^0}\left(1 - \frac{\tau^0}{\overline{\tau}}\right), \tag{55}$$

which we call the stability ratio. The average timescale, $\overline{\tau}$, is given by

$$\frac{1}{\overline{\tau}} \equiv \frac{1}{2}\left(\frac{1}{\tau^0} + \frac{1}{\tau^p}\right) \tag{56}$$

for the initial value τ^0 and the predicted value τ^p. Stability requires $R \leq 2$ for any species integrated with the asymptotic method [19]. If the timescale is constant or decreasing, this stability constraint is satisfied. If the timescale is increasing, Δt may be large enough to make the method unstable.

During the first 200 seconds of the simulation (i.e., the span of time shown in Fig. 7), CHEMEQ treats only O_2^- with the asymptotic update, so R for this species is included in the figure. The values of Δt are read from the vertical axis to the left of the figure, and the values of R correspond to the axis on the right. The stability limit of $R = 2$ is marked by a dashed horizontal line. We see that R becomes larger than 2 after approximately 10 s, and CHEMEQ starts producing oscillations in Δt approximately 10 s after that. These oscillations cease after R returns to values lower than 2. CHEMEQ2 does not produce these oscillations, although the accuracy-based timestep constraint lowers the timestep in this region.

A series of studies evaluated the accuracy of CHEMEQ2 compared to CHEMEQ. These solved the Cs test problem given above and used the reference solution at 1000 s as a benchmark. The tests varied the value of ϵ from 10^{-1} to 10^{-6}. Additional tests fixed ϵ and varied N_c from one to ten. Figure 8 summarizes the results of the tests by showing the rms error as a function of CPU time, which was scaled by the smallest increment the timing routine could resolve. The CHEMEQ2 results are shown as a series of overlapping profiles of the shape shown in the schematic in Figure 9. Each profile is for a fixed value of ϵ, but the points on it correspond to different values of N_c.

The error computed for each computation (fixed ϵ and N_c) is based on the the accepted values at 1000 s. The relative error e_i for each species i

$$e_i = \frac{y_{i,\text{accepted}} - y_{i,\text{calculated}}}{y_{i,\text{accepted}}}. \tag{57}$$

A root-mean-square error for the six reacting species (excluding the inert N_2) is:

$$e_{rms} = \sqrt{\frac{\sum_{i=1}^{6} e_i^2}{6}}. \tag{58}$$

There is only a single curve for CHEMEQ in Figure 5. Each point on this curve corresponds to a different value of ε. The hybrid method, as implemented in CHEMEQ and used in this problem, becomes unstable and the solutions are corrupted if multiple corrector iterations are used. Lorenzini and Passoni, however, were able to use multiple corrector iterations successfully in other implementations of the hybrid method for other problems [29]. CHEMEQ2 does not have this instability problem.

For a single-iteration and large enough ε, the CHEMEQ2 results lie roughly along the CHEMEQ curve. In this case, the CHEMEQ2 simulation takes less time, but gives a slightly less accurate solution. As ε is

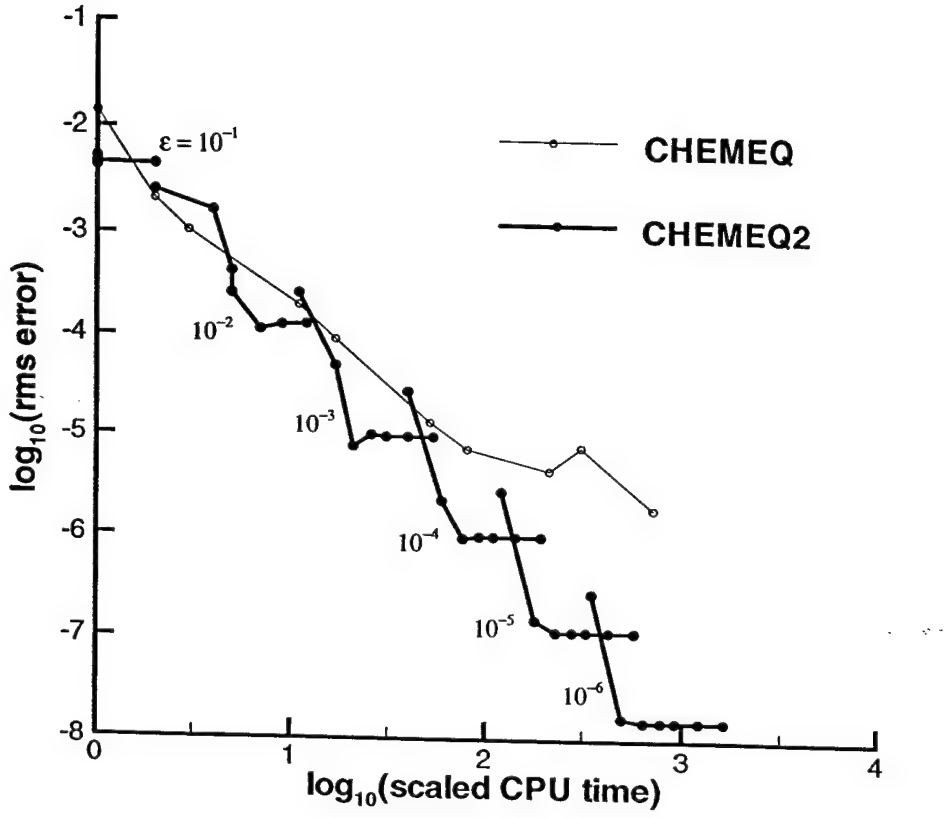

Figure 8: Root mean square error at 1000 s versus scaled CPU time to reach 1000 s for CHEMEQ and CHEMEQ2 for a range of ε and N_c. A schematic of each CHEMEQ2 curve is shown in Fig. 9, and the numbers next to each CHEMEQ2 curve indicate the value of ε for those results. The CHEMEQ results correspond to $\varepsilon = 10^{-1}$ (highest point on the curve) to 10^{-6} (lowest point).

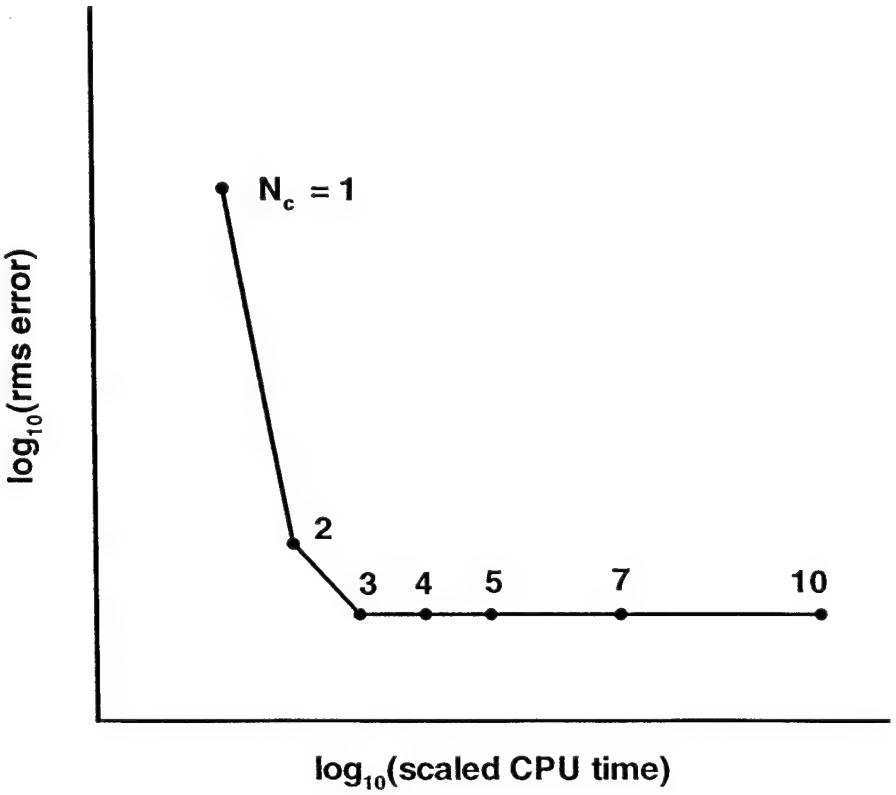

Figure 9: Schematic of the types of profiles for fixed ε in Fig. 8. The numbers next to each symbol give the corresponding value of N_c. As described in the text, the solutions for the cesium test problem converge after about three iterations.

decreased, the CHEMEQ results do not give the same increase in accuracy for the increased computational costs.

The curves shown in Figure 5 can be explained by comparing the CHEMEQ and the CHEMEQ2 algorithms. The CHEMEQ stiff predictor is identical to the CHEMEQ2 predictor in the limit as $\alpha^0 \rightarrow 1$, which corresponds to $p^0 \Delta t \rightarrow \infty$. The CHEMEQ stiff corrector, however, uses different average values for q and p than the CHEMEQ2 corrector, and also effectively uses the $p\Delta t \rightarrow 0$ limit value of $\alpha = 0.5$. This inconsistency in the effective α between CHEMEQ's stiff predictor and corrector limits the growth of Δt for CHEMEQ. Therefore, CHEMEQ takes a smaller timestep than CHEMEQ2 for the same ε, and, for moderate accuracy, this inconsistency in α does not affect the accuracy of the solution. The best accuracy achievable by CHEMEQ does suffer from this inconsistency, however, so as ε becomes smaller, CHEMEQ2 gives more accurate answers than CHEMEQ.

The CHEMEQ2 curves for a fixed ε show dramatic increases in accuracy after just a few iterations. After about three iterations, the curves for a given ε flatten, which indicates that the method has converged to a final corrector value, and additional iterations do not improve the accuracy. The computational expense in adding iterations is less than that in reducing ε for similar improvements in accuracy. As ε is lowered, accuracy improves because the timestep is decreased. As the number of iterations increases, accuracy improves because the corrector is able to refine the linear approximation for p and q used to calculate $\widetilde{q}$ and $\overline{p}$ for the corrector equation, Eq. (36). Not all systems will converge for such low values of N_c, but, in general, iterating the corrector even one or two times improves the accuracy.

For the CHEMEQ2 curve for $\varepsilon = 0.1$, the simulations took so little time that the precision of the timing routines was not sufficient to measure differences in timing between these runs. In addition, the calculations were performed on a computer that allows access to multiple users. These affects contribute to the error and uncertainty in the low-resolution data.

6.2 Hydrogen-Air Tests

The H_2-air combustion mechanism used consists of twenty-five reversible reactions involving nine species (including inert N_2) [34]. This mechanism is closely related to that used by GRIMech. The reaction rates

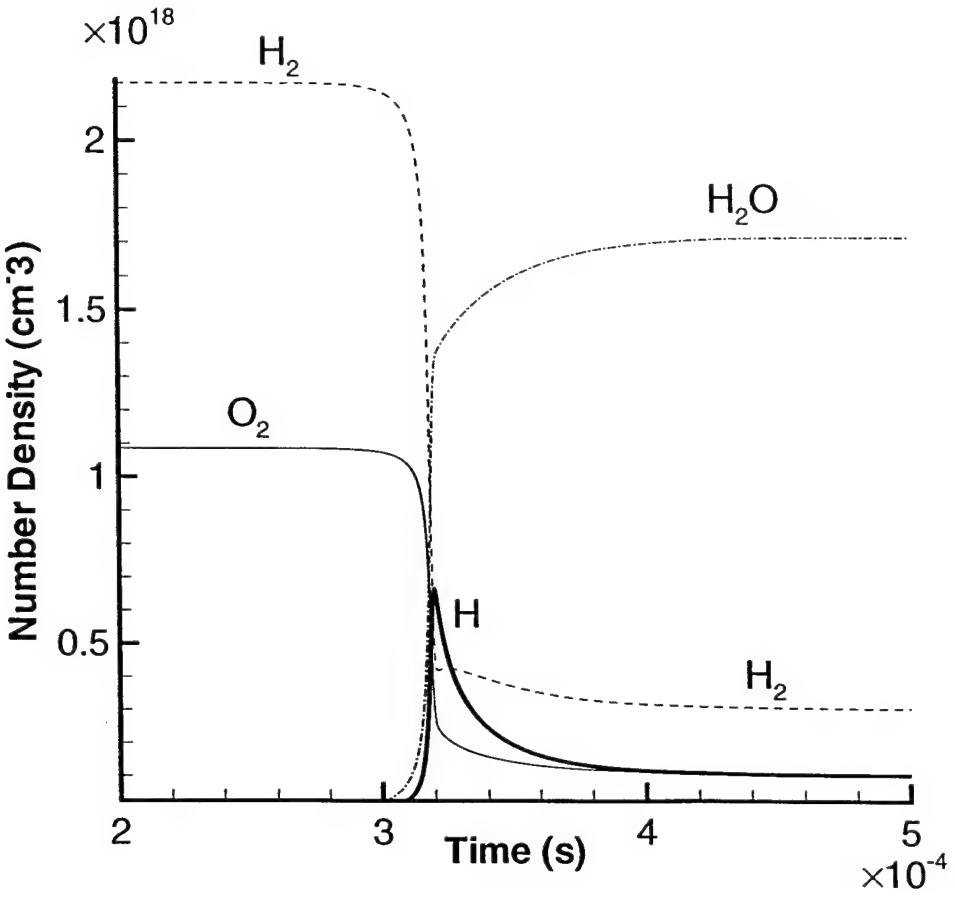

Figure 10: Solution for the single-point hydrogen-air integration.

are calculated using the modified Arrhenius form

$$k_r = AT^B \exp(-C/T) \tag{59}$$

where T is the temperature. The rate k_r is either a forward or backward rate. The parameters A, B, and C for both the forward and backward rates for each reaction are given in reference [19]. Initially the mixture is at 1000 K, a pressure of 1 atmosphere, and in the ratio 2:1:3.76 for H_2:O_2:N_2. These conditions lead to initial number densities on the order of 10^{18}cm^{-3} for these three species. A minimum number density of 10^{-30}cm^{-3} was imposed on the other species to prevent numerical difficulties. Nitrogen is inert for the mechanism, and thus acts as a diluent.

Selected species' number densities for this problem are presented as a function of time in Fig. 10. The figure shows that after an induction time of about 3.4×10^{-4} s, H_2 and O_2 are converted to H_2O in a

relatively short time period. During this induction time radicals are formed that eventually initiate the rapid conversion of H_2 and O_2. Here, we focus on the H number density profile, which has a peak in the reaction zone that is difficult to predict accurately. A series of calculations examined the effect of ε and N_c on the location and the value of this peak. The errors in these parameters are calculated as

$$t_p \text{ error} = \frac{\|t_p - t_{p,\text{reference}}\|}{t_{p,\text{reference}}} \tag{60}$$

$$(n_H)_p \text{ error} = \frac{\|(n_H)_p - (n_H)_{p,\text{reference}}\|}{(n_H)_{p,\text{reference}}} \tag{61}$$

for peak number density value $(n_H)_p$ at time t_p, compared to reference values. The reference values were obtained by integrating the equations with CHEMEQ2 for increasing N_c and decreasing ε values, until the solution ceased changing. The solution was then verified by comparison with a solution obtained by a simple modified Euler method using an exceptionally low error tolerance. Table 3 lists these errors and the CPU time required to reach 5×10^{-4} seconds for a variety of ε and N_c values. These calculations were performed on a DEC Alpha workstation, and the integration time is scaled relative to the $\varepsilon = 10^{-3}$, $N_c = 1$ simulation. These calculations did not assume that the thermodynamic state or the rate constants remained fixed during a chemical timestep. The temperature was recalculated for each corrector iteration based on the species number densities calculated from the previous iteration. The rate constants were then recalculated with this updated temperature. Repeating this calculation for every iteration is very expensive, and the performace of the method will improve substantially if the rate constants are calculated once and fixed for the chemical timestep.

Figure 11 shows results of integrations for $\varepsilon = 10^{-4}$ and $N_c = 1$, 5, and 10. This should be contrasted to the cesium calculations of the previous section that converged by $N_c = 5$. In this case, the profiles are converging to the reference solution, but they have not completely converged by $N_c = 10$. Note that CHEMEQ results are essentially equivalent to the $N_c = 1$ case. Table 3 suggests that reducing ε may be a more efficient way to improve the accuracy of the solution than increasing the iteration count. The errors are not of the same order as ε, however, and reducing ε by an order of magnitude does not result in a comparable reduction in the error. The errors in the time-to-peak and the peak value are not even comparable, with the

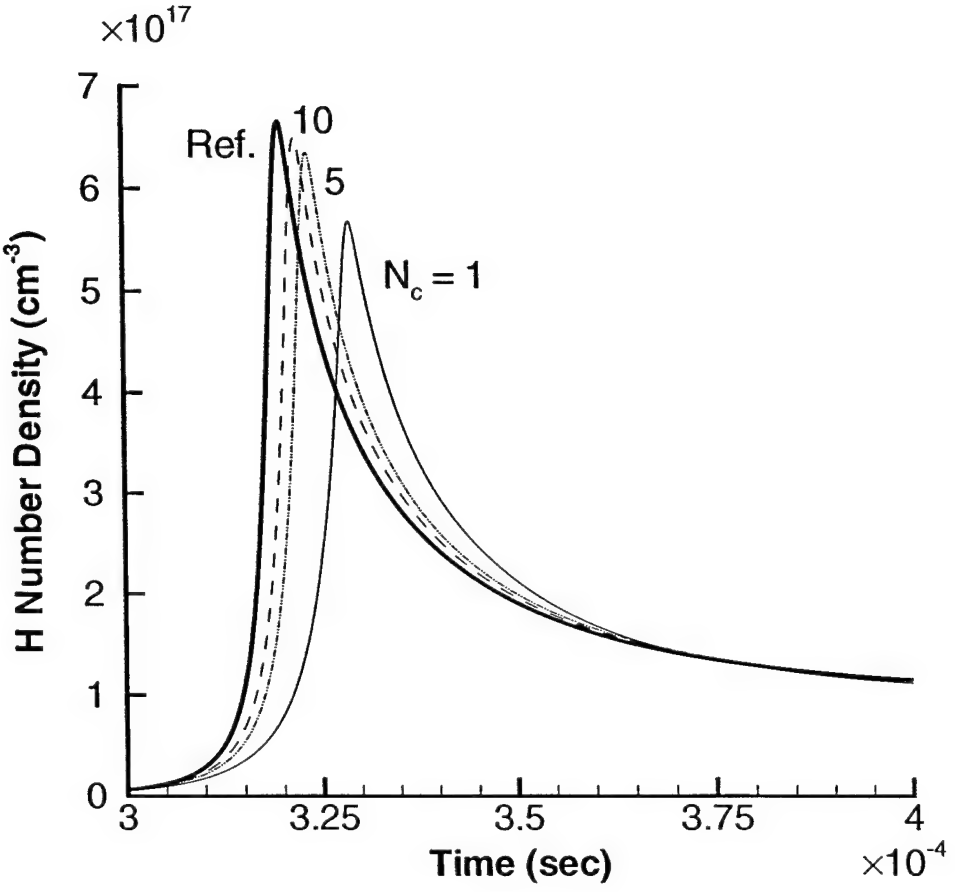

Figure 11: Hydrogen number density for $N_c = 1$, 5, and 10, and $\varepsilon = 10^{-4}$. The dark, solid line is the reference solution, and the numbers next to the remaining curves indicate the value of N_c for each profile.

Table 3: Results obtained by varying ε and N_c for the hydrogen-air reaction integration.

3(a): t_p error

ε	$N_c = 1$	5	10
10^{-3}	6.66×10^{-2}	2.97×10^{-2}	1.84×10^{-2}
10^{-4}	2.79×10^{-2}	1.10×10^{-2}	6.29×10^{-3}
10^{-5}	1.06×10^{-2}	3.67×10^{-3}	1.97×10^{-3}

3(b): $(n_H)_p$ error

ε	$N_c = 1$	5	10
10^{-3}	0.392	0.166	9.40×10^{-2}
10^{-4}	0.146	4.56×10^{-2}	2.35×10^{-2}
10^{-5}	4.48×10^{-2}	1.27×10^{-2}	6.49×10^{-3}

3(c): Scaled CPU times to 5×10^{-4} s.

ε	$N_c = 1$	5	10
10^{-3}	1.00	2.92	5.33
10^{-4}	3.19	9.92	18.3
10^{-5}	11.8	36.7	67.5

Table 4: Errors in t_p and $(n_H)_p$ for $\varepsilon = 10^{-3}$ and $N_c = 1$, 5, 10, and 1000, and the scaled CPU time required for each simulation to reach $t = 5 \times 10^{-4}$ seconds.

iterations	t_p error	y_H error	CPU time
1	6.66×10^{-2}	0.392	1.00
5	2.97×10^{-2}	0.166	2.92
10	1.84×10^{-2}	9.40×10^{-2}	5.33
1000	2.71×10^{-6}	1.77×10^{-4}	489

$(n_H)_p$ much more prone to error than t_p. This peak is very difficult for a low-order method to calculate. A higher-order method that employs information from several timesteps would provide a much better result for this problem.

The question remains as to how accurate the integration can become if the number of iterations is increased dramatically. The results for $\varepsilon = 10^{-3}$ from Table 3 are repeated in Table 4, and additional results obtained using 1000 corrector iterations are also included. For the $N_c = 1000$, the error in the peak value is an order of magnitude less than ε, and the time-to-peak error is three orders of magnitude lower than ε. This suggests that the corrector equation Eq. (36) provides an accurate representation once it is sufficiently converged. Again, the thermodynamic state and the rate constants were recalculated for every corrector iteration, and the high CPU times are due in part to this largely avoidable expense.

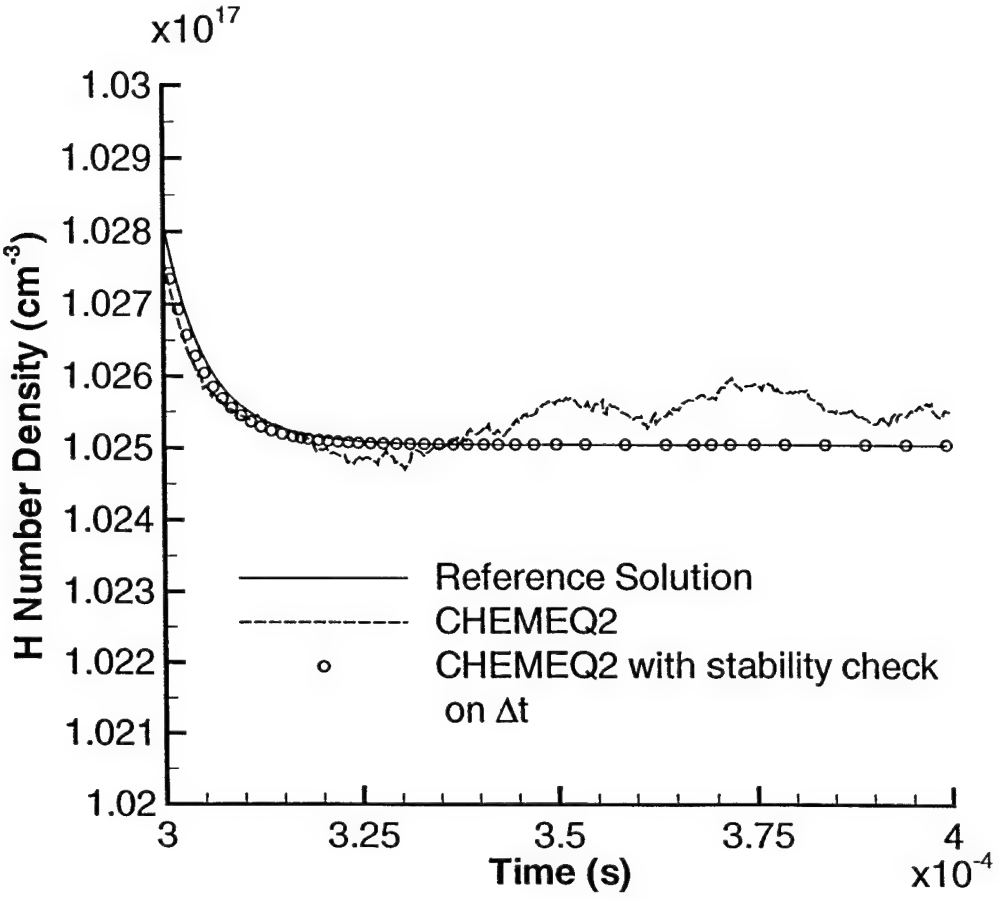

Figure 12: Hydrogen number density as the integration approaches equilibrium, $\varepsilon = 10^{-5}$, $N_c = 10$. The dashed line is the standard CHEMEQ2 result. The profile given by open circles includes the stability constraint on Δt (see Eq. (52)).

An instance which requires the stability check on Δt described in Section 4.2 is illustrated in Fig. 12. This figure shows the H profile as the system reaches equilibrium. These results indicate that the accuracy-based timestep can be too large for the corrector iteration to remain stable despite the fact that the stability analysis indicated that α-QSS is A-stable for linear problems. Note that the scale in Fig. 12 is exaggerated; the range covered by the number density axis spans approximately 1% of the equilibrium value. This instability is not a problem in reacting-flow applications, as the frequent restarting at new global timesteps limits how large the timestep becomes. In this single-point integration, however, the instability is seen. The oscillations in the number density disappear when the stability constraint given in Eq. (52) is required, and the predicted equilibrium value agrees well with the reference solution.

6.3 Reacting-Flow Solutions

Two reacting-flow cases will be briefly discussed here. These results are provisional, as no rigorous, systematic studies have been performed. A thorough comparison between integration methods would include the effects of implementation choices, accuracy requirements, and stiffness. The stiffness issues are not limited to the chemical mechanism itself but also include coupling of the chemical timescales and the fluid dynamics timescales (i.e., how much the integrator subdivides the global timestep in order to perform the chemistry integration). Such a study is planned for the future. However, from our experiences, we expect the results described below to be typical.

Uphoff et al. [35] studied two-dimensional detonation formation using an H_2/O_2 mechanism with 18 reactions and 8 species. They compared process-split reacting-flow calculations using CHEMEQ and METAN1 [36] as the chemistry integrator. METAN1 is a general stiff solver which employs a semi-implicit mid-point rule and extrapolation to a "zero stepsize" solution [37–39]. For this specific set of calculations, CHEMEQ performed the required chemical integrations in approximately one-sixth of the time required by METAN1. Documentation of accuracy parameters used and solution options chosen for the calculations is not available.

An additional calculation was performed in order to compare the efficiency of α-QSS to a Gear method. A one-dimensional hydrogen-air premixed flame was simulated using a process-split method [33] which employed FCT for integrating the fluid convection [40]. The chemistry integration was performed using CHEMEQ2, and also using DEBDF, which employs a variable-order Gear method as implemented in LSODE. DEBDF is part of SLATEC, a library of computational subroutines available on Silicon Graphics and Cray computers [41]. CHEMEQ performed the required calculations in approximately one-sixth the time required by DEBDF, which is coincidentally the same factor seen in the detonation comparison versus METAN1. No extensive accuracy studies have been performed to ensure that the comparison was fair. For example, the accuracy parameters for CHEMEQ2 and DEBDF were simply set to the same value, even though the two codes do not use these parameters in exactly the same way.

7 Summary

CHEMEQ2 is a general purpose integrator for a *specific type* of equations, namely those that are reasonably represented by the form in Eq. (1). CHEMEQ2 employs a very low overhead, moderately accurate, low-order technique. To obtain results for most physical models with an acceptable degree of accuracy, CHEMEQ2 can be extremely efficient. In many areas where problems are so computationally expensive they seem impossible to do by other methods, CHEMEQ2 gives accurate results in a reasonable amount of time. CHEMEQ2 can also be employed in the development of chemical or mathematical models when efficiency is important, but obtaining very precise answers may require extensive computational expense. CHEMEQ2 is optimized to provide three or four significant digits accurately, not eight, but this high level of accuracy can be reached with an appropriate timestep criterion and enough corrector iterations.

CHEMEQ2's forte lies in the solution of the stiff ordinary differential equations associated with chemically reactive flow problems. Here the reaction sources are split off from the hydrodynamic part of the equations and solved separately for each hydrodynamic timestep and at each grid point. The moderate accuracy of the methods used to solve the hydrodynamic equations suggest that the application of a more sophisticated technique, rather than a low-order, low overhead method like CHEMEQ2, would waste valuable computer time and could possibly render the problem impossible.

A potential user must be aware that CHEMEQ2 is not completely user-proof or problem-independent and cannot always be used as a black box. The method is not identically conservative, and the minimum values should be chosen with some thought since they can become sources of spurious errors if not chosen small enough initially. Although CHEMEQ2 overcomes some stability problems in the original algorithm, it may still require the use of the stability check described in Section 4.2. The user is referred again to Section 5.4 for practical information regarding the use of CHEMEQ2.

Since CHEMEQ2 uses a convergence-dependent algorithm and an adaptive timestep, the overall timing will be strictly problem-dependent. One factor will be the coupling between the relaxation times of the equations. The most expensive operation in the algorithm is the derivative function evaluations, of which there is one required in the predictor step, and one for each corrector iteration. If CHEMEQ2 is applied as

designed, the subroutine can solve large systems of stiff ordinary differential equations very efficiently.

8 Acknowledgements

This work was funded in part by the Office of Naval Research, and in part by a Department of Defense High Performance Computing Modernization Office (HPCMO) Common High Performance Computing Software Support Initiative (CHSSI) program within the Computational Fluid Dynamics Core Technology Area. It was completed while D. Mott was an NRC/NRL Postdoctoral Research Associate. The authors would like to thank Prof. Bram van Leer for his guidance and insight during the course of this research. The authors would also like to thank T.R. Young, Jr. and J.P. Boris, who developed the asymptotic method in CHEMEQ, for their help and advice, as well as G. Patnaik, who streamlined CHEMEQ for workstation use and performed the hydrogen-air flame calculations discussed in Section 6.3.

References

[1] T. R. Young and J. P. Boris, "A Numerical Technique for Solving Ordinary Differential Equations Associated with the Chemical Kinetics of Reactive-Flow Problems." *J. Physical Chemistry*, 81, 2424 (1977).

[2] T.R. Young, Jr., "CHEMEQ — A Subroutine For Solving Stiff Ordinary Differential Equations." NRL Memorandum Report No.4091 (1980).

[3] J.D. Hoffman, *Numerical Methods for Engineers and Scientists.*McGraw-Hill, Inc., New York, 1992.

[4] J.D. Lambert, *Numerical Methods for Ordinary Differential Systems; The Initial Value Problem.* John Wiley & Sons, Chichester, England, 1991.

[5] E.S. Oran, J.P. Boris, *Numerical Simulation of Reactive Flow.* Elsevier Science Publishing Co., Inc., New York, 1987.

[6] E.S. Oran, T.R. Young, J.P. Boris. "Application of Time-Dependent Numerical Methods to the Description of Reactive Shocks." *Proceedings of the 17th Symposium (International) on Combustion*, p. 43, the Combustion Institute, Pittsburgh, PA, 1979.

[7] K. Kailasanath, E.S. Oran, J.P. Boris. "A Theoretical Study of the Ignition of Premixed Gasses." *Combustion and Flame*, 47, 173, 1982.

[8] E.S. Oran, T.R. Young, J.P. Boris, and A. Cohen. "Weak and Strong Ignition: I. Numerical Simulations of Shock Tube Experiments." *Combustion and Flame*, 48, pp. 135–148, 1982.

[9] E.S. Oran and J.P. Boris. "Weak and Strong Ignition: II. Sensitivity of the Hydrogen-Oxygan System." *Combustion and Flame*, 48, pp. 149–161, 1982.

[10] K. Kailasasanth and E. S. Oran. "Ignition of Flamelets behind Incident Shock Waves and the Transition to Detonation." *Combustion Science and Technology*, 34, 345–361, 1983.

[11] G. A. Doschek, J.P. Boris, C.-C. Cheng, J.T. Mariska, and E.S. Oran. "Numerical Simulation of Cooling Coronal Flare Plasma." *The Astrophysical Journal*, 258, 373, 1982.

[12] C.-C. Cheng, E.S. Oran, G. A. Doschek, J.P. Boris, and J.T. Mariska. "Numerical Simulations of Loops Heated to Solar Flare Temperatures: I. Gasdynamics." *The Astrophysical Journal*, 265, 1090–1102, 1983.

[13] C.-C. Cheng, E.S. Oran, G. A. Doschek, J.P. Boris, and J.T. Mariska. "Numerical Simulations of Loops Heated to Solar Flare Temperatures: II. X-Ray and UV Spectroscopy." *The Astrophysical Journal*, 265, 1103–1119, 1983.

[14] J.W. Weber, Jr., E.S. Oran, J.D. Anderson, Jr., and G. Patnaik. "Load Balancing and Performance Issues for the Data Parallel Simulation of Stiff Chemical Nonequilibrium Flow." *AIAA Journal*, 35, pp. 147–163, 1998.

[15] E. S. Oran, J. W. Weber, Jr., E. I. Stefaniw, M. H. Lefebvre, and J. D. Anderson, Jr. "A Numerical Study of a Two-Dimensional H_2-O_2-Ar Detonation Using a Detailed Chemical Reaction Model." *Combustion and Flame* 113, pp. 147-163, 1998.

[16] G. Patnaik and K. Kailasanath. "Numerical Predictions of the Cell-Split Limit in Lean Premixed Hydrogen-Air Flames in Microgravity," presented at the *Joint Meeting of the United States Sections of the Combustion Institute*, Washington, D.C., March 1999.

[17] K. Kailasanath, G. Patnaik, and C. Li. "Computational Studies of Pulse Detonation Engines: A Status Report," AIAA 99-2634, and presented at the 35th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, June 1999.

[18] G. Patnaik and K. Kailasanath, "Outflow Boundary Conditions for Pulse Detonation Engines," presented at the 8th SIAM Conference on Numerical Combustion, March 2000.

[19] D.R. Mott. "New Quasi-Steady-State and Partial-Equilibrium Methods for Integrating Chemically Reacting Systems." Ph.D. Thesis, The University of Michigan, April 1999.

[20] D.R. Mott, E.S. Oran, and B. van Leer. "A Quasi-Steady-State Solver for the Stiff Ordinary Differential Equations of Reaction Kinetics," *Journal of Computational Physics*, 164, pp. 407–428, (2000).

[21] L. O. Jay, A. Sandu, F. A. Porta, and G. R. Carmichael, "Improved Quasi-Steady-State-Approximation Methods for Atmospheric Chamistry Integration." *SIAM Journal of Scientific Computing*, Vol. 18, No. 1, pp. 182 – 202 (1997).

[22] J. G. Verwer and D. Simpson, "Explicit methods for stiff ODEs from atmospheric chemistry." *Applied Numerical Mathematics*, 18, pp. 413 – 430 (1995).

[23] J. G. Verwer and M. van Loon, "An evaluation of explicit Psuedo-Steady-State Approximation Schemes for Stiff ODE Systems from Chemical Kinetics." *Journal of Computational Physics*, 113, 347 – 352 (1994).

[24] K. Radhakrishnan and D. T. Pratt. "Fast Algorithm for calculating Chemical Kinetics in Turbulent Reacting Flow," *Combustion Science and Technology*, Vol. 58, pp. 155-176, 1988.

[25] A. C. Hindmarsh. "LSODE and LSODEI, Two New Initial Value Ordinary Differential Equation Solvers." ACM SIGNUM Newsletter, vol. 15, no. 4, 1980, pp. 10-11.

[26] A. C. Hindmarsh. "ODEPACK: A Systemized Collection of ODE Solvers." *Scientific Computing*. R. S. Stepleman, et al., eds., North Holland Publishing Company, Amsterdam, 1983, pp. 55-64. (Also UCRL-880-07, Lawrence Livermore Laboratory, Livermore, CA, 1993).

[27] C.W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations* (Prentice Hall, Englewood Cliffs, New Jersey, 1971).

[28] K. Radhakrishnan. "New Integration Techniques for Chemical Kinetic Rate Equations. I. Efficiency Comparison," *Combustion Science and Technology*, Vol. 46, pp. 59-81, 1986.

[29] R. Lorenzini and L. Passoni. "Test of numerical methods for the integration of kinetic equations in tropospheric chemistry." *Computer Physics Communications* 117 (1999), pp. 241-249.

[30] C. J. Avro. "CHEMSODE: a stiff ODE solver for the equations of chemical kinetics." *Computer Physics Communications* 97 (1996), pp. 304-314.

[31] F. A. Williams, *Combustion Theory, The Fundamental Theory of Chemically Reacting Systems*, 2nd edition, Benjamin/Cummings Publishing Company, Menlo Park, California, 1985.

[32] S. H. Lam and D. A. Goussis, "The CSP Method for Simplifying Kinetics." *International Journal of Chemical Kinetics*, 26, pp. 461 – 486, 1994.

[33] G. Patnaik, K.J.Laskey, K. Kailasanath, E.S. Oran, and T.A. Brun, "FLIC — A Detailed, Two-Dimensional Flame Model." NRL Memorandum Report No.6555 (1989).

[34] M. Frenklach, H. Wang, and M. J. Rabinowitz, Optimization and Analysis of Large Chemical Kinetic Mechanisms Using the Solution Mapping Method — Combustion of Methane, *Progress in Energy and Combustion Science* .18, pp. 47–73 (1992).

[35] U. Uphoff, D. Hänel, P. Roth, A Grid Refinement Study for Detonation Simulation with Detailed Chemistry, Proc. of 6th Int. Conf. on Num. Combustion, New Orleans, March 4-6, 1996.

[36] P. Deuflhard, U. Nowak, and U. Poehle. Scientific Software Group, Konrad-Zuse-Zentrum fuer Informationstechnik Berlin, 1989.

[37] P. Deuflhard, A Semi-Implicit Midpoint Rule for Stiff Systems of Ordinary Differential Equations, *Numeriche Mathematik* **41**, 373 - 398, 1983.

[38] P. Deuflhard, Order and Stepsize Control in Extrapolation Methods, *Numeriche Mathematic* **41** (1983), 399-422.

[39] P. Deuflhard, Uniqueness Theorems for Stiff ODE Initial Value Problems, Konrad-Zuse-Zentrum fuer Informationstechnik Berlin, Preprint SC-87-3 (1987).

[40] J. P. Boris, Alexandra M. Landsberg, Elaine S. Oran, and John H. Gardner. "LCPFCT — A Flux-Corrected Transport Algorithm for Solving Generalized Continuity Equations." Memorandum Report 6410-93-7192, Naval Research Laboratory, 1993.

[41] W. H. Vandevender and K. H. Haskell, The SLATEC mathematical subroutine library, SIGNUM Newsletter, 17, 3 (September 1982), pp. 16-21.

9 Appendix A – CHEMEQ2 Variable List

Table 9: Variable listing and descriptions.

FORTRAN variable	Type/Scope	Same As	Description
alpha	R/L	α, Eq. (7)	solution parameter used in update
ascr	R/L	—	scratch (temporary) variable
d(i)	R/A	$p_i y_i$, Eq. (1)	loss rate
dt	R/L	Δt	chemical timestep used by the integrator
dto	R/L	—	stores timestep; used to scale rtaus when timestep is reduced
dtc	R/L	$\varepsilon y_i / g_i$	diagnostic value printed when $\Delta t <$ dtmin
dtg	R/A	Δt_g	global timestep; range of integration
dtmin	R/L	—	minimum timestep allowed
dtmn	R/A	—	sets dtmin via CHEMSP
epscl	R/L	1/epsmin	intermediate variable used to avoid repeated divisions
eps	R/L	σ, Eq. (48)	maximum correction term, finally scaled by 1/epsmin
epsmax	R/A	c from Eq. (47)	repeat timestep if correction is greater than epsmax*epsmin*y(i) for any i
epsmin	R/A	ε from Eq. (46)	accuracy parameter for determining the next timestep
epsmn	R/A	—	used to set epsmin via CHEMSP
epsmx	R/A	—	used to set epsmax via CHEMSP
gcount	I/L	—	counter for calls to gsub since the last call to CHEMCT
gsub	E/A	—	source term subroutine; supplies d(i) and q(i)
i	I/L	i	index
iter	I/L	—	counter for corrector iterations
itermax	I/A	—	number of corrector iterations to perform
itermx	R/A	—	used to set itermax via CHEMSP
lo	I/L	—	unit number for output
n	I/A	n	number of equations integrated
nd	I/L	—	dimension of species arrays; maximum number of species
ns	I/A	—	number of entries in ymin reset via CHEMSP call
prt	R/A	—	nonzero value supresses output from CHEMSP
q(i)	R/A	q_i, Eq. (1)	production rate
qs(i)	R/A	q_i^0, Eqs. (35), (38)	initial production rate
qt	R/A	$\widetilde{q}_i$, Eq. (38)	α-weighted average of q
rcount	I/L	—	counter for steps redone since the last call to CHEMCT

Type: R = Real, I = Integer, E = External; Scope: L = Strictly Local, A = Passed as Argument

Table 9 Continued

FORTRAN variable	Type/Scope	Same As	Description
rswitch	R/L	5.9659	value of $\Delta t/\tau$ used to switch between Eqs.(39) and (41) when Padé (a) is used
rtau(i)	R/L	$\Delta t/\tau_i$	ratio of timestep to timescale
rtaub	R/L	$\overline{p_i}\Delta t = \Delta t/\overline{\tau_i}$	Δt times average p from Eq. (37)
rtaui	R/L	$\Delta t/\tau_i$	holds rtau(i) to avoid multiple array references
rtaus(i)	R/L	$\Delta t/\tau_i^0$	ratio of timestep to initial timescale for current timestep
rteps	R/L	$\sqrt{\sigma}^*$	estimate for $\sqrt{\sigma}$ in Eq. (50)
scr1	R/L	—	scratch (temporary) variable
scr2	R/L	—	scratch (temporary) variable
scrarray	R/L	—	scratch (temporary) variable array
scrtch	R/L	—	scratch (temporary) variable
sqreps	R/A	$5\sqrt{\varepsilon}$	parameter used to calculate initial timestep
stab	R/L	—	$\|\Delta y_i^{c(N_c-1)}\|/\|\Delta y_i^{c(N_c)}$; see Eqs. 52) and (53)
tfd	R/L	—	round-off parameter used to determine when integration is complete
tgcnt	I/L	—	total number of calls to gsub for all global timesteps
tmk	R/A	—	call identifier for CHEMCT
tn	R/A	$t - t^0$	current value of the independent variable relative to the start of the global timestep
tnot	R/A	—	used to set tstart via CHEMSP
trcnt	I/L	—	total number of steps redone for all global timesteps
ts	R/A	—	independent variable at the start of the chemical timestep
tstart	R/A	t^0	independent variable at the start of the *global* timestep
y(i)	R/A	y_i	species concentrations array
y0(i)	R/A	y_i^0 from Eq. (2)	initial concentrations for the *global* timestep passed to CHEMEQ
y1(i)	R/A	y_i^p	predicted value from Eq. (35)
ym1(i)	R/L	$y_i^{c(l-1)}$	previous corrector iterate; see Eq. (51)
ym2(i)	R/L	$y_i^{c(l-2)}$	previous corrector iterate; see Eq. (51)
ymin(i)	R/L	—	minimum concentration allowed for species i
ymin(i)	R/A	—	set ymin(i) via CHEMSP
ys(i)	R/L	y_i^0 from Eqs. (35) and (36)	initial concentrations for the chemical timestep

Type: R = Real, I = Integer, E = External; Scope: L = Strictly Local, A = Passed as Argument

10 Appendix B: Code Listings

10.1 CHEMEQ2 Code Listing

```
      subroutine chemeq2(dtg, gsub, n, y)
c
cd* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
cd
cd    chemeq2(dtg, gsub, n, y)
cd
cd    original chemeq development:
cd    originators:  t.r. young                                  nrl 1982
cd    vax version:  t.r. young            nrl code 4040         may 1983
cd    workstation:  g. patnaik            berkeley research     jun 1995
cd
cd    chemeq2 development: d.r. mott      nrl code 6404         may 1999
cd
cd
cd    Description:  Subroutine chemeq2 solves a class of "stiff" ODEs
cd    associated with reactive flow problems that cannot be readily
cd    solved by the standard classical methods. In contrast to the
cd    original chemeq subroutine, this version uses the same
cd    quasi-steady-state update for every species regardless of the
cd    timescale for  that species. An adaptive stepsize is  chosen to
cd    give accurate results for the fastest changing quantity, and a
cd    stability check on the timestep is also available when the
cd    corrector is iterated.
cd
cd    NOTE:  The accuracy-based timestep calculation can be augmented
cd    with a stability-based check when at least three corrector
cd    iterations are performed.  To include this check, "uncomment"
cd    the lines that start with "D", or use the compiler flag "-d_lines"
cd    if available to compile the code including these lines.  If the
cd    lines are manually uncommented,  the continuation characters
cd    must be placed in the correct column.  For most problems, the
cd    stability check is not needed, and eliminating the calculations
cd    and logic associated with the check enhances performance.
cd
cd    The routine assumes that all of the integrated quantites and the
cd    time step are positive.
cd
cd    argument list definition (name, type, description, input vs. output):
cd    dtg             real        the interval of integration or the   i
cd                                range of the independent variable.
cd                                0.0 <= t <= dtg. (global timestep)
cd    gsub            real        the name of the derivitive function  i
cd                                evaluator subroutine.
cd    n               integer     the number of equations to be        i
cd                                integrated. an error exisis if n is
cd                                greater than nd set by the parameter
cd                                statement.
cd    y(n)            real        the initial values at call time    i/o
cd                                and the final values at return time.
```

```
cd
cd    Language and limitations:  This subroutine is written in standard
cd    FORTRAN 77.  For high accuracy, this routine should be compiled
cd    using whatever "double precision" flag is appropriate for the
cd    platform being used (such as "f77 -r8 . . . .")
cd
cd    Entry points:  Four entry points are provided for flexibility and
cd    optimum control.  This structure was maintained from the original
cd    chemeq subroutine to ensure compatiability with previous
cd    applications that use chemeq.
cd
cd    chemeq2:  advances the equations the given increment 'dtg'.
cd
cd    chemct:  informative, prints the values of the indicative
cd             counters listed below;
cd             1.  the number of derivative function evaluations.
cd             2.  the number times the integration step was restarted
cd                 due to nonconvergence of the predictor-corrector
cd                 scheme.
cd
cd    chemsp:  provides the user with the option to reset the most
cd             important control parameters.
cd
cd    chempr:  informative, prints out internal variables for diagnostic
cd             purposes.
cd
cd    subroutines referenced:
cd
cd    gsub;    whose actual name and definition are supplied by the user
cd             is called to obtain the derivitive functions.
cd
cd    call gsub(y, q, d, t)
cd    argument list to gsub;
cd    y(n)              real      current values of the dependent      i
cd                                variable.
cd    q(n)              real      calculated formation rates.          o
cd    d(n)              real      calculated loss rates.               o
cd    t                 real      current value of the independent     i
cd                                variable.
cd
cd    chemer: Called whenever an error is detected. Currently the
cd            only error recognized is a time step that is too small.
cd
cd    call chemer(y, n)
cd    argument list to chemer;   (same definition as "chemeq2").
cd
cd* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
c
      implicit none
      integer nd
      parameter    (nd = 10)
c
```

```
      external gsub
      integer      n, ns, lo, i
      integer      itermax, iter, itermx
c
c     the following are counters (this call & total) for gsub calls
c      and timestep repeats
c
      integer      gcount,  rcount, tgcnt, trcnt
c
      real         ts, tn, tfd, tmk
      real         y(n)
      real         ymin(nd), ymn(nd)
      real         q(nd), d(nd), rtaus(nd), y1(nd)
      real         ys(nd), y0(nd), rtau(nd)
      real         alpha, qs(nd)
      real         scr1, scr2, scrarray(nd)
      real         epscl, dtg, dtmin, sqreps, tstart, dt, dto
      real         epsmax, epsmin, rswitch
      real         epsmx, epsmn, dtmn, tnot, prt
      real         scrtch, ascr, eps
      real         rtaui, rtaub, qt, pb, dtc, rteps
c
c   ym1, ym2, and stab are used only for the stability check on dt
D     real         ym1(nd), ym2(nd), stab
c
      data         gcount,  rcount, tgcnt, trcnt/4*0/
      data          itermax/1/, epscl/100.0/
      data         tfd/1.000008/, dtmin/1.0e-15/, sqreps/0.50/
      data         tstart, dt/2*0.0/, tn/0.0e+00/, q/nd*0.0/
      data         epsmax/10.0/, lo/16/, epsmin/1.0e-02/, d/nd*0.0/
      data         rswitch/  5.965900 /
c
c rswitch for 4-4 pade: 5.9659
c
cd     check input parameters.
          if(n .gt. nd) then
             write(lo, 1002) n, nd
 1002     format(5(/),'from -chemeq2-  :  no. of eq.s requested is too',
     .     ' large'/' requested (',i5,'), max. allowed (',i5,')')
             stop
          end if
c
c     initialize the control parameters.
  110     tn = 0.0e+00
c
c     store and limit to 'ymin' the initial values.
          do i = 1, n
             q(i) = 0.0
             d(i) = 0.0
             y0(i) = y(i)
             y(i) = max(y(i), ymin(i))
          end do
```

```
c     evaluate the derivitives of the initial values.

          call gsub(y, q, d, tn + tstart)
          gcount = gcount + 1
c
c     estimate the initial stepsize.
c
c     strongly increasing functions(q >>> d assumed here) use a step-
c     size estimate proportional to the step needed for the function to
c     reach equilibrium where as functions decreasing or in equilibrium
c     use a stepsize estimate directly proportional to the character-
c     istic stepsize of the function. convergence of the integration
c     scheme is likely since the smallest estimate is chosen for the
c     initial stepsize.
          scrtch = 1.0e-25
          do  i = 1, n
             ascr = abs(q(i))
             scr2 = sign(1./y(i),.1*epsmin*ascr - d(i) )
             scr1 = scr2 * d(i)
             scrtch = max(scr1,-abs(ascr-d(i))*scr2,scrtch)
          end do
          dt = min(sqreps/scrtch,dtg)
c
c     the starting values are stored.
 100      ts = tn
c
          do i=1,n
             rtau(i) = dt*d(i)/y(i)
             ys(i) = y(i)
             qs(i) = q(i)
             rtaus(i) = rtau(i)
          end do
c
c
c     find the predictor terms.
  101 continue
c
      do i = 1,n
c
c     prediction
c
         rtaui = rtau(i)
c
c        note that one of two approximations for alpha is chosen:
c        1) Pade b for all rtaui (see supporting memo report)
c             or
c        2) Pade a for rtaui<=rswitch,
c           linear approximation for rtaui > rswitch
c           (again, see supporting NRL memo report (Mott et al., 2000) )
c
c   Option 1): Pade b
```

```
c
        alpha = (180.+rtaui*(60.+rtaui*(11.+rtaui)))
     &        / (360. + rtaui*(60. + rtaui*(12. + rtaui)))
c
c   Option 2): Pade a or linear
c
c         if(rtaui.le.rswitch) then
c            alpha = (840.+rtaui*(140.+rtaui*(20.+rtaui)))
c     &           / (1680. + 40. * rtaui*rtaui)
c         else
c            alpha = 1.-1./rtaui
c         end if
c
         scrarray(i) = (q(i)-d(i))/(1.0 + alpha*rtaui)
      end do
c
      iter = 1
      do while(iter.le.itermax)

c     limit decreasing functions to their minimum values.
         do i = 1,n
D            ym2(i) = ym1(i)
D            ym1(i) = y(i)
            y(i) = max(ys(i) + dt*scrarray(i), ymin(i))
         end do

         if(iter.eq.1) then
c
c  the first corrector step advances the time (tentatively) and
c   saves the initial predictor value as y1 for the timestep check later
            tn = ts + dt
            do i=1,n
               y1(i) = y(i)
            end do
         end if
c
c     evaluate the derivitives for the corrector.
c
         call gsub(y, q, d, tn + tstart)
         gcount = gcount + 1
         eps = 1.0e-10

         do i = 1,n
c
            rtaub = .5*(rtaus(i)+dt*d(i)/y(i))
c
c  Same options for calculating alpha as in predictor:
c
c  Option 1): Pade b
c
               alpha = (180.+rtaub*(60.+rtaub*(11.+rtaub)))
     &              / (360. + rtaub*(60. + rtaub*(12. + rtaub)))
```

```
c  Option 2):  Pade a or linear
c
c             if(rtaub.le.rswitch) then
c                alpha = (840.+rtaub*(140.+rtaub*(20.+rtaub)))
c     &               / (1680. + 40.*rtaub*rtaub)
c             else
c                alpha = 1.-1./rtaub
c             end if

              qt = qs(i)*(1. - alpha) + q(i)*alpha
              pb = rtaub/dt
              scrarray(i) = (qt - ys(i)*pb) / (1.0 + alpha*rtaub)
c
           end do
c
           iter = iter + 1
c
        end do
c
c     calculate new f, check for convergence, and limit decreasing
c     functions. the order of the operations in this loop is important.
        do i = 1,n
           scr2 = max(ys(i) + dt*scrarray(i), 0.0)
           scr1 = abs(scr2 - y1(i))
           y(i) = max(scr2, ymin(i))
D          ym2(i) = ym1(i)
D          ym1(i) = y(i)
c
           if(.25*(ys(i) + y(i)).gt.ymin(i)) then
              scr1 = scr1/y(i)
              eps = max(.5*(scr1+
     &             min(abs(q(i)-d(i))/(q(i)+d(i)+1.0e-30),scr1)),eps)
c
           end if
        end do
        eps = eps*epscl
c
c     print out dianostics if stepsize becomes too small.

        if(dt .le. dtmin + 1.0e-16*tn) then
           write(lo, 1003) dt, tn, dtmin
           do i = 1,n
              dtc = epsmin*y(i)/(abs(q(i)-d(i)) + 1.0e-30)
              write(lo, 1004) q(i), d(i), y(i), rtau(i), dtc,
     &             q(i)-d(i),ys(i), y0(i), ymin(i)
           end do

 1003     format('1    chemeq error;   stepsize too small ! ! !', /,
     1    '     dt = ', 1pe10.3, ' tn = ', d25.15,
     2    ' dtmin = ',e10.3, //, 11x, 'q', 10x, 'd', 10x, 'y',
     3         8x, 'rtau', 8x, 'dtc', 7x, 'q - d',7x, 'ys',
```

```
     4         9x, 'y0', 8x, 'ymin')
 1004     format(5x, 1p12e11.3)
             dt = dtg - ts
             dt = min(dtmin, abs(dt))
c
c          call error diagnostic routine
             call chemer
c
          end if
c
c          check for convergence.
c
c  The following section is used for the stability check
D           stab = 0.01
D           if(itermax.ge.3) then
D              do i=1,n
D                 stab = max(stab, abs(y(i)-ym1(i))/
D     &                 (abs(ym1(i)-ym2(i))+1.e-20*y(i)))
D              end do
D           endif

        if(eps .le. epsmax
D     &    .and.stab.le.1.
     &          ) then
c
c       Valid step.  Return if dtg has been reached.
c
           if(dtg .le. tn*tfd) return
        else
c
c       Invalid step; reset tn to ts
c
           tn = ts
        end if
C
c       perform stepsize modifications.
c       estimate sqrt(eps) by newton iteration.
c
        rteps = 0.5*(eps + 1.0)
        rteps = 0.5*(rteps + eps/rteps)
        rteps = 0.5*(rteps + eps/rteps)
c
        dto = dt
        dt = min(dt*(1.0/rteps + .005), tfd*(dtg - tn)
D     &       ,dto/(stab+.001)
     &       )
c
c       begin new step if previous step converged.
c
        if(eps .gt. epsmax
D     &       .or. stab. gt. 1
     &       ) then
```

```
          rcount = rcount + 1
c
c         After an unsuccessful step the initial timescales don't
c         change, but dt does, requiring rtaus to be scaled by the
c         ratio of the new and old timesteps.
c
          dto = dt/dto
          i = 1
          do while(i.le.n)
             rtaus(i) = rtaus(i)*dto
             i = i+1
          end do
c
c         Unsuccessful steps return to line 101 so that the initial
c         source terms do not get recalculated.
c
          goto 101
        end if
c
c       Successful step; get the source terms for the next step
c       and continue back at line 100
c
        call gsub(y, q, d,tn + tstart)
        gcount = gcount + 1
        go to 100
c
c
c
        entry chemct (tmk)
c      ----- ------
c
cd* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
cd
cd    chemct (tmk)
cd    write out the values of the various indicative counters that the
cd    program keeps.
cd
cd    argument list definition:
cd    tmk               real      a floating point number printed      i
cd                                to identify the call.
cd
cd    output variable definition:
cd    tmk               real      floating point identifier.
cd    gcount            integer   number of derivative subroutine calls
cd                                since the last call.
cd    rcount            integer   number of times stepsize was reduced
cd                                since last call.
cd    tgcnt             integer   total of gcount to this call.
cd    trcnt             integer   total of rcount to this call.
cd
cd* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
c
```

```
          tgcnt = tgcnt + gcount
          trcnt = trcnt + rcount
c
c     print out indicative counters.
          write(lo, 1000) tmk, gcount, rcount, tgcnt,
     .     trcnt
 1000     format(' chemeq indices; tmk = ', 1pe10.3,
     .     ' gcount, rcount = ', 2i7, ' totals: ', 2i7)
c
c     reset counters.
          gcount = 0
          rcount = 0
          return
c
c
c
      entry chemsp(epsmn, epsmx, dtmn, tnot, itermx, ns, ymn, prt)
c     ----- ------
c
cd* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
cd
cd    chemsp(epsmn, epsmx, dtmn, tnot, itermx, prt)
cd
cd    reset any local control parameters if their respective input
cd    values are greater than zero. default values are used if the
cd    input values are zero or less repectively.
cd
cd    argument list definition:
cd    epsmn             real      the maximum relative error allowed   i
cd                                for convergence of the corrector step.
cd                                default value:  1.0e-02
cd    epsmx             real      this number provides the basis for   i
cd                                deciding weather convergence can be
cd                                achieved with out added stepsize
cd                                reduction. if eps/epsmin is greater
cd                                than epsmx further reduction is
cd                                applied.
cd                                default value : 10.0
cd    dtmn              real      the smallest stepsize allowed.       i
cd                                default value:  1.0e-15
cd    tnot              real      the initial value of the independent i
cd                                variable t.
cd                                default value:  0.0
cd    itermx            i         number of times the corrector is applied
cd                                default value:  1
cd    ns                integer   number of entries in ymin to reset   i
cd
cd    ymn(nd)           real      minimum values allowed for y         i
cd                                default value: 1.0e-20
cd    prt               real      controls the output of chemsp.  any  i
cd                                non zero value suppresses all print
cd                                output from this entry.
```

```
cd
cd* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
c
          epsmin = 1.0e-02
          if(epsmn .gt. 0.0)epsmin = epsmn
          if(epsmn .gt. 0.0)sqreps = 5.0*sqrt(epsmin)
          epscl = 1.0/epsmin
          epsmax = 10.0
          if(epsmx .gt. 0.0)epsmax = epsmx
          dtmin = 1.0e-15
          if(dtmn .gt. 0.0)dtmin = dtmn
          tstart = tnot
          itermax = 1
          if(itermx.gt. 0) itermax = itermx
          do i=1,ns
             ymin(i) = 1.e-20
             if(ymn(i).gt.0.) ymin(i) = ymn(i)
          end do
c
c     print new values of control parameters.
          if(prt .eq. 0.0) then
             write(lo, 1001) epsmn, epsmx, dtmn, tnot, itermx
             write(lo, 1005) ns
             if (ns.gt.0) write(lo,1006) (ymin(i), i=1,ns)
          end if

 1001     format(' initalize "chemeq2" via "chemsp"', /,
     .    ' epsmn, epsmx, dtmn, tnot, itermx = ', 1p5g10.3)
 1005     format(' ns = ',I5)
 1006     format(' ymin: ',50e12.3)

          return
c
c
c
      entry chempr (y, n)
c     ----- ------
c
cd* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
cd
cd    chempr (y, n)
cd
cd    chempr may be called whenever an error occurs that can be
cd    attributed to the results of chemeq. a partial set of the internal
cd    variables is printed as a diagnostic.
cd
cd    argument list definition:
cd    y(n)         r  current values of the dependent variable.     i
cd    n            i  the number of entries in y and ymin.          i
cd
cd* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
c
```

```
      write(lo, 1003) dt, tn, dtmin
      do 45 i = 1,n
      dtc = epsmin*y(i)/(abs(q(i) - d(i)) + 1.0e-30)
45    write(lo, 1004) q(i), d(i), y(i), rtau(i),
     &     dtc, q(i)-d(i),  ys(i), y0(i), ymin(i)
c
      return
      end
      subroutine chemer
c     ---------- ------
c
c     diagnostic routine for stiff o.d.e. solver  -chemeq-
c
      print 1001
1001  format(5(/), ' library version of -chemer- called.', /,
     .     ' users may supply their own version for diagnostics.', /,
     .     ' no arguments are required.', /,
     .     ' program will continue resetting the step size to min-', /,
     .     'imums if a normal return is made.', //,
     .     ' (stop 69) executed from library version of -chemer-')
c
      stop 69
      end
```

10.2 Example Driver Code and Source Term Subroutines

```
      PROGRAM CESIUM
C     ------- -----
C
C     This is the driver program for the seven-species cesium
C     mechanism test problem.  The code integrates the system
C     MXCASE times using differnt values of the chemeq2 variable
C     epsmin (set by passing an entry from array EPS through
C     CHEMSP before each integration).
C
C     PROGRAM SPECIFICATIONS.
C     -----------------------
      REAL          DSEC
C
      REAL          Y(10), YF(10), YMIN(10), YI(10), EPSIL(10), EPS(15)
C
      INTEGER       SPSYM(7)
C
C     For this example, the external subroutine that calculates the
C     source terms is called CSDFE.
C
      EXTERNAL      CSDFE
C
      DATA          YMIN/10*1.0E-20/, MXCASE/9/, LO/16/
      DATA          SPSYM/'O2-', 'CS+', 'CS', 'CSO2', 'O2', 'N2', 'NE'/
      DATA          EPS/
     .              .1, .05, .01, .005,
     .              .001, .0005, .0001, .00005,
     .              .00001, .000005, .000001,
     .              5.e-7, 1.e-7,5.e-8,1.e-8
C     .               ,5.e-9, 1.e-9,5.e-10,1.e-10
     .              /
C
 1000     FORMAT('CASE NO. ', I5, '    PARAMETERS;', /,
     .    ' CONVERGENCE PARAMETER EPS = ', 1PE10.3, /,
     .    ' INNER LOOP LENGTH;', I5)
 1001     FORMAT(/, '    SPECIE    Y - INITAL      Y - FINAL ',
     .    '  Y - SOLUTION   REL ERR')
 1002     FORMAT(5X, A4, 1P3E15.6, E10.3)
 1003     FORMAT(/, ' T - INITIAL = (', 1PE10.3, ') T - FINAL = (',
     .    E10.3, ')')
 1004     FORMAT(/' INTEGRATION STATISTICS;')
 1005     FORMAT(' CPU TIME USED FOR INTEGRATION;', 1PE10.3,
     .    ' SEC.,  CPU TIME NORMALIZED;', I8)
 1006     FORMAT(' SUM OF THE RELATIVE ERRORS SQUARED; ', 1PE10.3)
 1007     FORMAT(/)
C
C         Note that the timing routines included may not work on
C         all systems.  Extra timing options are included as comments.
C
          REAL*4 dtime, delta, tarray(2)
```

```
      integer tnorm
      EXTERNAL dtime
      delta = 1.
C
C
C     INITIALIZE CONTROL PARAMETERS.
C
C         TSCALE is simply a normalization factor for the timing
C         results.  It can be used to compare results from differnt
C         machines (by setting it to the time required for that
C         machine to solve a standard problem of some sort) or to
C         simply make the timing results more "friendly."
          TSCALE = 1.0/1024.
C
C         INLP allows the user to subdivide the interval over which
C         each test is run.  For INLP=1, CHEMEQ2 is sent the full
C         interval TF-TI (specified below) as the global timestep.
          INLP = 1
C
C
C     For this particular test, the electron number density is not
C     integrated.  The other five reacting species are integrated,
C     and the electron density is found through charge conservation.
C     This calculation is done within CSDFE.  Therefore, NA = 5 is
C     the number of equations that are integrated, but NS = 7 is the
C     number of species.  Species to be integrated must be placed in
C     first NA positions within the Y array.  CHEMEQ2 only works with
C     these first NA entries since NA is passed in the argument list
C     below, but all NS values are available to and used by CSDFE.
C
          NS = 7
          NA = 5
C
C     "TI"  -  INITIAL TIME,   "TF"  -  FINAL TIME.
          TI = 0.0
          TF = 1000.0
          DELTAT = (TF - TI)/INLP
C
C     STORE INITIAL(TI = 0.0) AND FINALT(F = 1000.0) VALUES.
C
C     O2-
          YI(1) = 5.200E+02
          YF(1) = 2.59139492061D+04
C
C     CS+
          YI(2) = 6.200E+02
          YF(2) = 7.55718460300D+04
C
C     CS
          YI(3) = 1.000E+12
          YF(3) = 1.53194051722D+03
C
```

```
C     CSO2
          YI(4) = 0.
          YF(4) = 9.99999923516D+11
C
C     O2
          YI(5) = 3.600E+14
          YF(5) = 3.59000000051D+14
C
C     N2
          YI(6) = 1.400E+15
          YF(6) = 1.40000000000D+15
C
C     NE
          YI(7) = 1.000E+02
          YF(7) = 4.96578968239D+04

C
C     LOOP OVER THE TEST CASES.
          DO 30 ICASE = 1, MXCASE
             WRITE(LO, 1000) ICASE, EPS(ICASE), INLP
             CALL CHEMSP(EPS(ICASE), 0., 0., TI, 5, ns, ymin, 0.)
             CPUT = 0.0
C
C     RESET "Y" TO INITIAL VALUES "YI".
          DO  I = 1,NS
             Y(I) = YI(I)
          end do
C
C     SET TIMER.
C          T1 = SECNDS(0.0)
          delta = dtime(tarray)

C
C     INNER LOOP TO DETERMINE OVERHEAD OR RELATIVE STARTING EFFECIENCY
C     OF ITEGRATION SCHEME BEING TESTED.
          DO ISTEP = 1,INLP
C
C     CALL INTEGRATOR.
             CALL CHEMEQ2(DELTAT, CSDFE, NA, Y)
C
          end do
C
C     CALCULATE CPU TIME USED IN THE INTEGRATION PROCESS.
          delta = dtime(tarray)
C          DSEC = SECNDS(T1)
          DSEC = tarray(1)
C          DSEC = delta
          CPUT = CPUT + DSEC
          TNORM = INT(CPUT/TSCALE + .5)
C     Calculate final electron density from densities of other charges species
          Y(7) = Y(2) - Y(1)
C
```

```
C     CALCULATE RELATIVE ERROR.
          DO  I = 1,NS
             EPSIL(I) = ABS(Y(I) - YF(I))/MIN(Y(I) , YF(I))
          end do
          SUM = 0.0
          DO I = 1,NS
             SUM = SUM + EPSIL(I)**2
          end do
C   Root-mean-square error is calculated using ns-1 (rather than ns)
C   since N2 is inert.
c
          sum = sqrt(sum/real(ns-1))
c
C
C     PRINT RESULTS.
          WRITE(LO, 1003) TI, TF
          WRITE(LO, 1001)
          DO 15 I = 1,NS
   15     WRITE(LO, 1002) SPSYM(I), YI(I), YF(I), Y(I), EPSIL(I)
          WRITE(LO, 1004)
          WRITE(LO, 1006) SUM
          WRITE(LO,  1005) CPUT, TNORM
          WRITE(*,699)  EPS(ICASE),
     &         CPUT, TNORM, sum
 699      format(1x,25HEPS, time, ticks, error: ,E7.1,2x,e10.4,2x,
     &         I5,2x,e10.4)
          WRITE(LO, 1007)
          CALL CHEMCT(TF)
   30     CONTINUE
          STOP 69
      END

      subroutine csdfe(y, q, d, t)
c
cd * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
cd
cd    csdfe(y, q, d, t)
cd
cd    description:
cd    derivative function evaluator(gsub) for an atmospheric chemical
cd    relaxation test problem involving cesium and cesium ions. format-
cd    ion and loss rates are calculated for this set of "stiff ordinary
cd    differential equations" that was suggested by by d. edelson of
cd    bell laboratories.
cd
cd    argument list definitions:
cd    y(i)          r     current values of the functions plus the      i/o
cd                        extra data at the end of the array that may be
cd                        passed back and forth between "csdfe" and the
cd                        main program. locations in y(i) which represent
cd                        the functions being advanced should not be
cd                        tampered with here.
```

```
cd    q(i)          r    total formation rates.                          i
cd    d(i)          r    total loss rates.                               i
cd    t             r    the value of the independent variable.          i
cd
cd * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
c
c             local specifications.
c             ---------------------
      real          ne, n2
      real          y(1), q(1), d(1)
c
c     utilize local storage for varibles.
          o2m  = y(1)
          csp  = y(2)
          cs   = y(3)
          cso2 = y(4)
          o2   = y(5)
          n2   = y(6)


c           write(63,*) t
c
c     calculate electron density for local use and transmission back to
c     the main program via y(7). however in this case this value should
c     not be trusted since "chemeq" will not call the "gsub" with the
c     latest function values after the final step has converged. y(7)
c     will be one iteration behind in this case. y(7) and y(6) are
c     examples tho, of how data may be transfered between the "gsub" and
c     the main program.
          ne = max(csp - o2m, 0.0)
          y(7) = ne
c
c     calculate reaction rates.
          cr1 = 5.00e-08*o2m*csp
          cr2 = 1.00e-12*csp*ne
          cr3 = 3.24e-03*cs
          cr4 = 4.00e-01*o2m
          cr5 = 1.00e-31*o2*cs*(cs + cso2 + n2 + o2)
          cr6 = 1.24e-30*o2*o2*ne
          cr7 = 1.00e-31*o2*n2*ne

c           if(t.ge.700.) then
c              cr4= 0.
c              cr6 = 0.
c              cr7 = 0.
c           end if

c
c     calculate total formation rates (c(i)) and total loss rates (d(i))
c     for each species.
c
c     o2m
```

```
          q(1) = cr6 + cr7
          d(1) = cr1 + cr4
c
c     cs+
          q(2) = cr3
          d(2) = cr1 + cr2
c
c     cs
          q(3) = cr1 + cr2
          d(3) = cr3 + cr5
c
c     cso2
          q(4) = cr5
c          q(4) = q(4) - 1.00e-31*o2*cs*cso2
c          d(4) = - 1.00e-31*o2*cs*cso2
c
c     o2
          q(5) = cr1 + cr4
          d(5) = cr5 + cr6 + cr7
c
          return
      end
```

10.3 Output from the Sample Programs

Running the cesium test problem as given with an "-r8" compiler flag (or equivalent) will result in the following screen output describing the case, unscaled and scaled run time, and the resulting rms error:

```
 EPS, time, ticks, error: 0.1E+00  0.9760E-03      1  0.4336E-02
 EPS, time, ticks, error: 0.5E-01  0.1952E-02      2  0.1035E-02
 EPS, time, ticks, error: 0.1E-01  0.5856E-02      6  0.1093E-03
 EPS, time, ticks, error: 0.5E-02  0.7808E-02      8  0.5876E-04
 EPS, time, ticks, error: 0.1E-02  0.2342E-01     24  0.9786E-05
 EPS, time, ticks, error: 0.5E-03  0.3611E-01     37  0.4845E-05
 EPS, time, ticks, error: 0.1E-03  0.8491E-01     87  0.9995E-06
 EPS, time, ticks, error: 0.5E-04  0.1200E+00    123  0.5195E-06
 EPS, time, ticks, error: 0.1E-04  0.2538E+00    260  0.1154E-06
69
```

Of course, run times will differ on different platforms, and the timing routines called by the driver routine may not be available on all systems.

Additional output found in `fort.16` is given below. This file holds a more detailed account of the results of each integration, including a count of the number of times the source term subroutine was called and the number of timesteps that were redone. Included below is this information for the last calculation in the test problem, for EPS = 1.000E-05.

```
CASE NO.      9    PARAMETERS;
 CONVERGENCE PARAMETER EPS =  1.000E-05
 INNER LOOP LENGTH;    1
 initalize ''chemeq2'' via ''chemsp''
 epsmn, epsmx, dtmn, tnot, itermx =  1.000E-05 0.000E+00 0.000E+00 0.000E+00
     5
 ns =     7
 ymin:    0.100E-19   0.100E-19   0.100E-19   0.100E-19   0.100E-19
 0.100E-19  0.100E-19

 T - INITIAL = ( 0.000E+00) T - FINAL = ( 1.000E+03)

    SPECIE    Y - INITAL      Y - FINAL   Y - SOLUTION   REL ERR
     O2-    5.200000E+02   2.591395E+04   2.591395E+04 9.164E-08
     CS+    6.200000E+02   7.557185E+04   7.557185E+04 9.755E-08
     CS     1.000000E+12   1.531941E+03   1.531941E+03 2.271E-07
     CSO2   0.000000E+00   9.999999E+11   9.999999E+11 1.467E-08
     O2     3.600000E+14   3.590000E+14   3.590000E+14 1.485E-09
     N2     1.400000E+15   1.400000E+15   1.400000E+15 0.000E+00
     NE     1.000000E+02   4.965790E+04   4.965790E+04 1.006E-07

 INTEGRATION STATISTICS;
 SUM OF THE RELATIVE ERRORS SQUARED;  1.154E-07
 CPU TIME USED FOR INTEGRATION; 2.538E-01 SEC.,  CPU TIME NORMALIZED;
 260

 chemeq indices; tmk =  1.000E+03 gcount, rcount =  149451      3 totals:
 313597     29
```